

TRATAMIENTO DE AUDIO PARA LA MEJORA DE LA CALIDAD DE VOZ EN PLATAFORMAS DE STREAMING

IMPROVING AUDIO QUALITY IN STREAMING APPLICATIONS



TRABAJO FIN DE GRADO
CURSO 2020-2021

AUTOR
EDUARDO ALCOBER CLEMENTE

DIRECTORES
MIGUEL GÓMEZ-ZAMALLOA & JAIME SÁNCHEZ HERNÁNDEZ

GRADO EN INGENIERÍA INFORMÁTICA
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

TRATAMIENTO DE AUDIO PARA LA MEJORA DE LA CALIDAD DE VOZ EN PLATAFORMAS DE STREAMING

IMPROVING AUDIO QUALITY IN STREAMING APPLICATIONS

TRABAJO DE FIN DE GRADO EN INGENIERÍA INFORMÁTICA
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS & COMPUTACIÓN

AUTOR

EDUARDO ALCOBER CLEMENTE

DIRECTORES

MIGUEL GOMEZ-ZAMALLOA & JAIME SÁNCHEZ HERNÁNDEZ

CONVOCATORIA: JUNIO 2021

GRADO EN INGENIERÍA INFORMÁTICA
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

RESUMEN

Este proyecto surge como una iniciativa, ante el aumento del uso de plataformas de emisión y el auge del teletrabajo, para proporcionar una herramienta fácil de usar que mejore la calidad de las emisiones. El objeto es el estudio de la inteligibilidad de la voz transmitida en streaming, en estos tiempos, donde las cifras de comunicaciones IP han alcanzado récords históricos (Encierros preventivos por el Covid-19), facilitando el uso mediante una interfaz simple pero configurable, a unos procesadores bastante complejos y en algunos casos desconocidos: los compresores de audio. La compresión de rango dinámico, inventada en la década de los 1930s-1940s, respondía a la necesidad del control automático del volumen de los presentadores de radio y televisión, debido a su variedad expresiva durante las emisiones en directo y para contener la potencia de la señal generada por toda una audiencia aplaudiendo a la vez. Hoy día, esta herramienta se sigue utilizando en formato hardware en todo tipo de estudios de grabación y emisiones, pero no aparece incorporada en los sistemas más habituales de streaming que tienden al uso de la Inteligencia Artificial en sus mejoras de audio, o no permiten la inclusión de procesadores software en la cadena de emisión. Mi objetivo es crear un plugin de audio, que se pueda usar en la mayor cantidad de entornos posible, incorpore una versión digital de las mejoras analógicas habitualmente aplicadas a la señal vocal, que se pueda autoconfigurar mediante presets, y finalmente estudiar el impacto de la compresión y el filtrado digital en la inteligibilidad de la transmisión.

Palabras clave

teletrabajo, streaming, compresión de rango dinámico, tratamiento de la señal, audio digital, VST2, inteligibilidad, filtros de audio, compresor, ecualizador

ABSTRACT

This Project arises as a response to the increase of the use of live broadcasting platforms and remote working to provide an easy-to-use tool which improves the quality of broadcasts. Its aim is the study of voice intelligibility on live streaming, in these times, where IP communication figures have reached historical records (due to Covid-19), facilitating the use through a simpler interface, but configurable to some quite complex and sometimes unknown processors: audio compressors. Dynamic Range Compression, invented during the 30s, responded to the need of automatic control of the volume of the TV & Radio hosts, due to their expressive variety while broadcasting live, as well as to contain the signal strength generated by a whole audience clapping at the same time. Nowadays, this tool is still used as hardware in all sorts of recording studios and streaming, but it is not implemented within the most usual streaming software, which tend to the use of AI on their audio improvements, or do not allow the inclusion of software processors within the broadcast chain. My aim is to create an audio plugin, that can be used on as many environments as possible, incorporating a digital version of the analogical improvements, usually applied on a vocal signal, that can be set up through presets, and lastly, to study the impact of compression and digital filtering on broadcasting intelligibility.

Key words:

telework, streaming, dynamic range compression, digital audio, signal processing, VST2, intelligibility, audio filters, compressor, equalizer.

ÍNDICE DE CONTENIDOS

Resumen.....	III
Abstract.....	IV
Índice de contenidos.....	V
Capítulo 1 - Introducción.....	1
Capítulo 1 - Introduction.....	2
1.1 Motivación	3
1.2 Objetivos.....	5
Capítulo 2 - Componentes y métricas de una señal de voz y su relación con la compresión	10
2.1 Introducción a la compresión de rango dinámico	10
2.2 Componentes frecuenciales audibles de la señal de voz	15
2.3 Plug ins incorporados en OBS	18
2.3.1 Compresor de rango dinámico	18
2.3.2 Limitador	20
2.4 Plug ins de Reaper	21
Capítulo 3 - La implementación en JUCE	23
El Procesador	24
El Editor de Audio.....	27
3.1 El compresor de rango dinámico	29
3.2 Filtro Paso Altos.....	30
3.3 La función de Auto-Threshold.....	33
Capítulo 4 - Los experimentos	35
4.1 Primer experimento: Objetivos y metodología	35

4.1.1 Objetivos	35
4.2 Primer experimento	37
4.2.1 Hipótesis	37
4.2.2 Método	37
4.2.3 Resultados.....	39
4.3 Segundo experimento.....	43
4.3.1 Hipótesis	44
4.3.2 Método	44
4.3.3 Resultados.....	44
Capítulo 5 - Conclusiones y trabajo futuro.....	49
Capítulo 6 - Conclusions and future work	51
Bibliografía.....	54
Índice de figuras	LV
Índice de tablas.....	LVII
Dedicatoria	LIX
Agradecimientos	LXI

Capítulo 1 - Introducción

Con la aparición de la Covid-19 a nivel mundial, y un mercado que no deja un hueco sin cubrir, son muchas las plataformas que permiten mantener reuniones a distancia en tiempo real y crear salas o equipos con un gran número de participantes, donde comienza la experiencia de trabajo colaborativa y la calidad de la comunicación juega un papel fundamental.

Excluidas las empresas cuya naturaleza no permite el trabajo a distancia, el 48,8% de las compañías aseguran que seguirán facilitando el teletrabajo en la nueva normalidad (Adecco, 2019-2020)

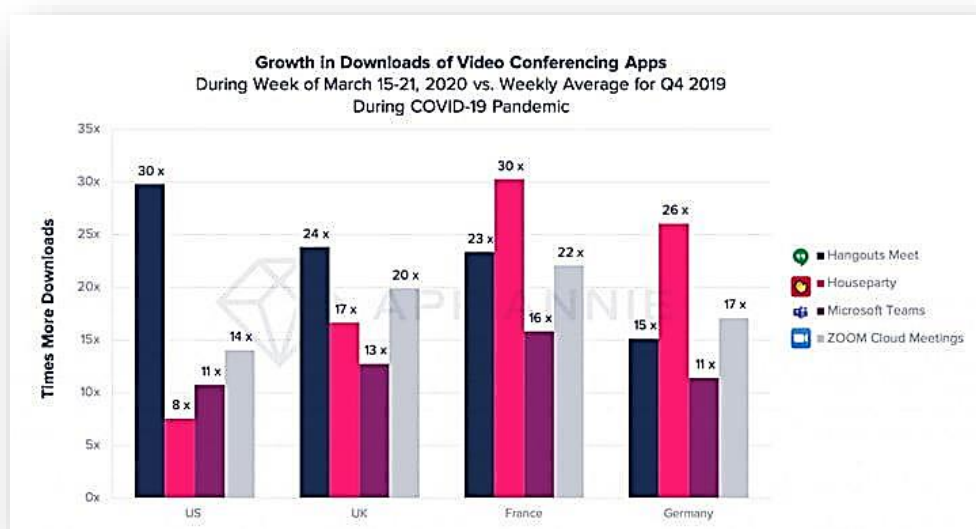


Ilustración 1, Índice de crecimiento de las descargas de aplicaciones de videoconferencia durante la semana del 15 al 21 de Marzo del 2020

Google Meet, Microsoft Teams y Zoom son las tres aplicaciones más utilizadas en los ámbitos educativo, empresarial y personal. Estos entornos proporcionan cada uno una serie de procesos propios no estandarizados para la mejora de la calidad de la señal de audio y su inteligibilidad. En la Tabla 1 podemos ver las opciones disponibles en las principales plataformas a este efecto, y en la Ilustración 1 el crecimiento en

descargas durante la semana del 15 al 21 de marzo del 2020, en los momentos de mayor incertidumbre respecto al Covid-19.

Durante este trabajo vamos a desarrollar un procesador de audio digital, basándonos en las herramientas tradicionales para la mejora de audio, y emitiendo a través de él, se estudiarán las variaciones provocadas en la inteligibilidad de la voz en función de los diferentes parámetros de configuración aplicados.

Capítulo 1 - Introduction

With the appearance of Covid-19 worldwide, and a market that does not leave a gap unfilled, there are many platforms that allow remote meetings to be held in real time and to create rooms or teams with a large number of participants, where it begins the collaborative work experience and the quality of communication play a fundamental role.

Excluding companies whose nature does not allow remote work, 48.8% of companies assure that they will continue to facilitate teleworking in the new normal (Adecco, 2019-2020)

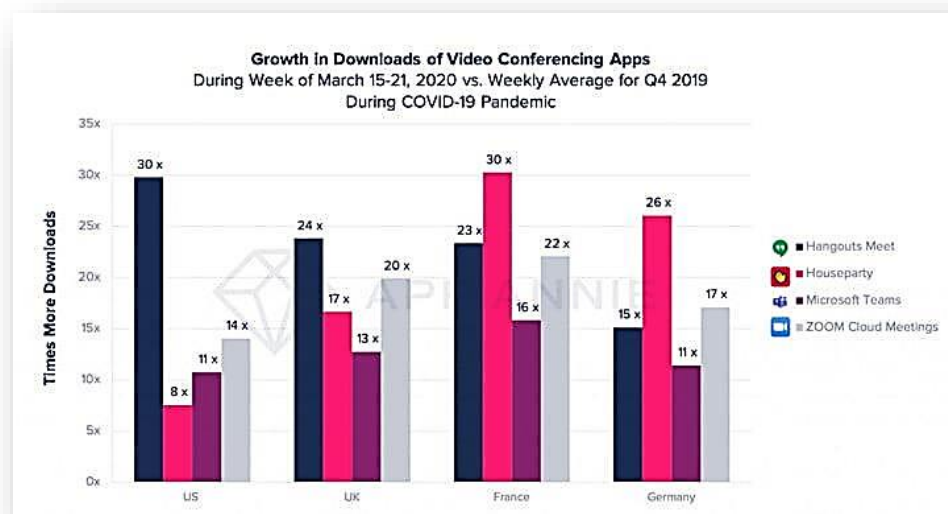


Illustration 1, Growth rate of videoconferencing applications downloads during the week of March 15-21, 2020

Google Meet, Microsoft Teams, and Zoom are the three most used apps in education, business, and personal life. These environments each provide a series of their own non-standardized processes for improving the quality of the audio signal and its intelligibility. In Table 1 we can see the options available on the main platforms for this purpose, and in Illustration 1 the growth in downloads during the week of March 15 to 21, 2020, at times of greatest uncertainty regarding Covid-19.

During this work we are going to develop a digital audio processor, based on traditional tools for audio improvement, and emitting through it, the variations caused in the intelligibility of the voice will be studied depending on the different configuration parameters applied.

1.1 Motivación

Estos sistemas de videoconferencia nombrados en el epígrafe anterior, son privados y de pago en sus versiones más completas. No son compatibles con plug-ins ni complementos que permitan mejoras más allá de aquellas introducidas en los roadmaps de los desarrolladores. No obstante, existe otro tipo de aplicaciones de streaming, que, satisfaciendo otro tipo de necesidades, permiten transmitir en directo desde nuestro PC convirtiéndolo en un estudio de producción y realización audiovisual, o simplemente de emisión "live". En muchas ocasiones son también utilizados para retransmitir en tiempo real la experiencia de usuario, con determinado software, principalmente de entretenimiento, como por ejemplo jugando a un juego de reciente lanzamiento. Este es el caso de la aplicación OBS u Open Broadcaster Software, que convierte nuestro ordenador personal en un estudio de streaming permitiendo un control exhaustivo de las fuentes durante la emisión.

OBS nos permite autenticarnos con Twitch y emitir a través de esta plataforma en nuestro canal. Permite además instalar y utilizar plugins externos VST, interfaz estándar desarrollada por Steinberg para extender las funcionalidades de los DAW (Digital Audio Workstation) con plugins externos, como sintetizadores y efectos de audio.

Esto permitiría en particular en OBS, insertar plugins procesadores de sonido para mejorar la calidad del audio, por ejemplo, para filtrar ruidos y normalizar volúmenes.

Streamlabs¹ realiza cada trimestre un informe de horas de visualización en cada una de las plataformas más importantes de este campo, que podemos ver en la Ilustración 2.

Tabla 1, mejoras de audio disponibles en plataformas de comunicaciones & teletrabajo

Plataforma	Mejoras disponibles de audio
Google Meet	<ul style="list-style-type: none">• Reducción / Cancelación de ruido automática y predeterminada
Microsoft Teams	<ul style="list-style-type: none">• Supresión de ruidos mediante IA configurable (automático, alto, bajo o desactivado)
Zoom	<ul style="list-style-type: none">• Suprimir ruido de fondo constante• Suprimir ruido de fondo intermitente• Cancelación de eco• Habilitar sonido original• Sonido de Alta fidelidad
Twitch	<ul style="list-style-type: none">• Compatible con OBS (Open Broadcaster Software)

¹ Software de streaming propietario de Logitech fundado en 2014 con sede en San Francisco y Vancouver (es.wikipedia.org, 2021)

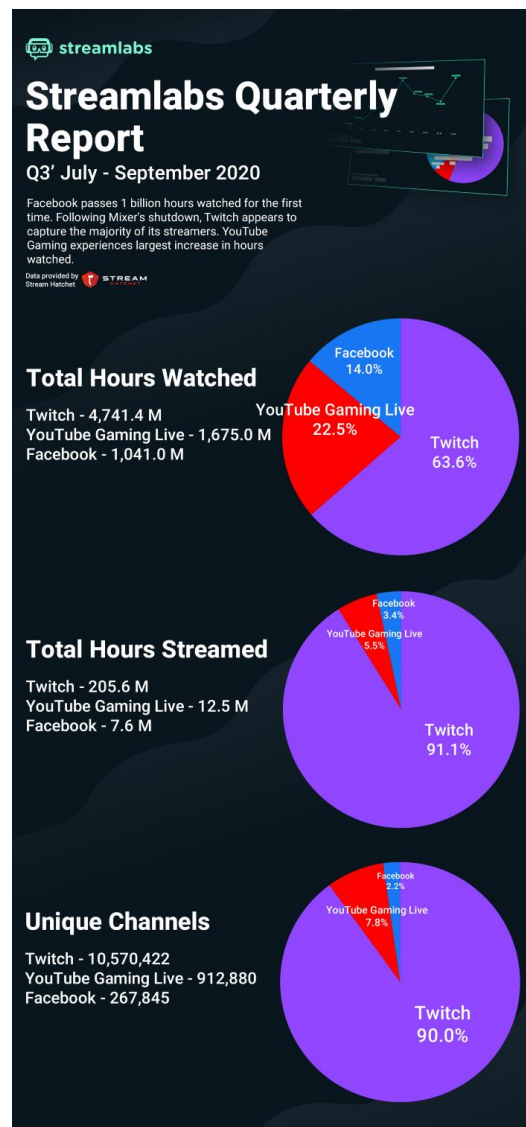
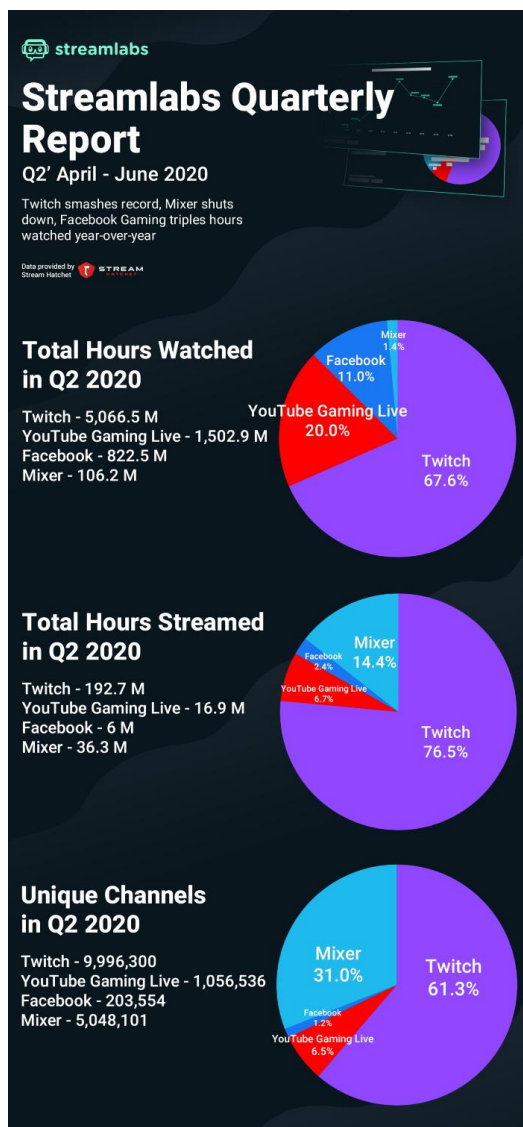


Ilustración 2. Resultado informe Streamlabs (forbes) 2020 Q2, Q3
(<https://streamlabs.com/content-hub/post/streamlabs-and-stream-hatchet-q3-2020-live-streaming-industry-report>)

En el tercer trimestre del 2020 hasta un 91,1% de las emisiones fueron servidas a través de la plataforma Twitch, por lo que será la utilizada para grabar las emisiones en este proyecto.

1.2 Objetivos

El propósito del presente TFG es el estudio de la mejora en la inteligibilidad de la señal de voz desde su emisión y en plataformas de streaming, así como el desarrollo de una herramienta a tal efecto, que sea fácil de utilizar para el público en general.

Entendemos como incremento de la calidad en el contexto del streaming, la mejora de la inteligibilidad de la voz, manteniendo una alta fidelidad a la señal original, y como *público en general* nos referimos a aquellos usuarios de nivel medio, con interés y capaces de configurar una sesión de emisión *live*.

Concretamente vamos a utilizar OBS en el que podemos insertar el plugin desarrollado durante este trabajo en la cadena de salida a emisión de audio. Emitiremos a través de TWITCH por ser la plataforma más utilizada, para registrar nuestras emisiones, y los servicios reconocedores de texto, o Speech to Text, de Microsoft y Google, para comprobar el acierto en función del texto reconocido en la señal emitida.

Tradicionalmente los mezcladores analógicos de estudio tienen una serie de secciones de procesamiento instaladas en cadena que se pueden habilitar y configurar en función de señal a tratar. Esto permite un control exhaustivo de la señal que los atraviesa.



Ilustración 3, Canal SSL 500

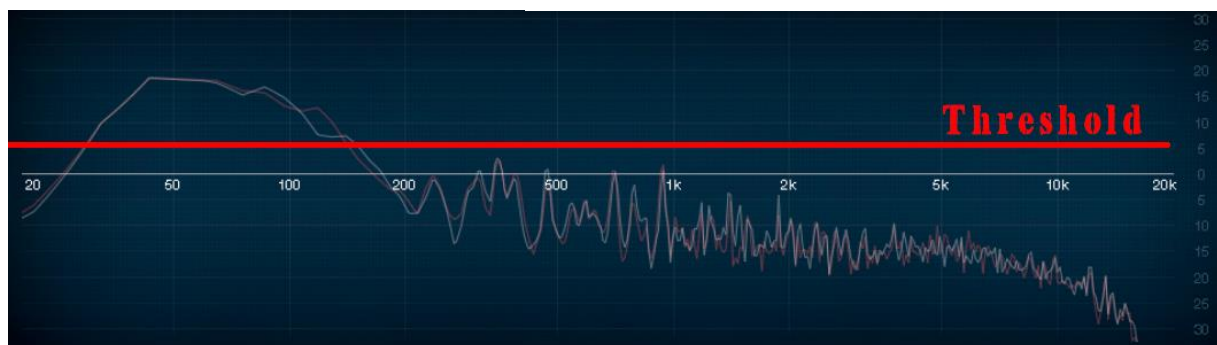
La *ecualización* y la *compresión del rango dinámico* de la señal que atraviesa el canal, son dos de las herramientas más versátiles y presentes en todos los modelos de mezcladores de gama media-alta. En la ilustración 3 podemos ver un canal extraído y convertido en módulo, de una mesa de mezclas de estudio con diferentes secciones.

A nuestro propósito, configurando un ecualizador como un filtro *paso altos* (*High Pass Filter* en inglés), eliminamos las componentes frecuenciales más graves carentes de información vocal anteriores a determinada frecuencia de corte, donde suelen alojarse ruidos no deseados en la señal: por eso en bastante literatura al respecto, escuchamos hablar de *filtros* y de *señal filtrada*. El filtro paso altos es un tipo de filtro con dos parámetros, que suelen llamarse frecuencia de corte y Q, o factor de calidad, que representa la pendiente a partir de la frecuencia de corte. Su función es eliminar las componentes de la señal desde 20 Hz (límite inferior del rango audible humano) hasta la frecuencia que configuremos. En nuestro caso configuraremos una frecuencia de corte de unos 250 Hz, eliminando las componentes graves, y permitiendo que pasen aquellas más cargadas de información.

Por otro lado, mediante un compresor de rango dinámico, podemos normalizar el volumen de la señal: estableciendo un umbral o límite correcto, hacemos que las componentes más altas de la señal que llegan o sobrepasan el umbral configurado, decrementen su volumen, mientras que el resto no se ven afectados, resultando ésta con un nivel medio más uniforme, rica armónicamente y más suave en su escucha. Estamos reduciendo su rango dinámico, y acortando la distancia en decibelios entre sus partes más altas y las más débiles, lo que produce una homogeneización del volumen de la pista tratada.

Vamos a implementar una cadena de procesamiento compuesta por un filtro paso altos y un Compresor de rango dinámico, de forma que la señal llegue filtrada y más limpia al compresor, permitiendo que ese exceso en frecuencias graves no sea el que active su funcionamiento al llegar al umbral, parámetro definido en el siguiente párrafo.

Ilustración 4, señal mal preparada para ser comprimida



En el siguiente capítulo trataremos más en detalle todas las opciones de trabajo del compresor. Por ahora nos interesa exponer que el umbral o *threshold* es el parámetro que marca el nivel de actuación del compresor, es decir, el volumen o amplitud que, al ser alcanzado por la señal, dispara la operación del compresor.

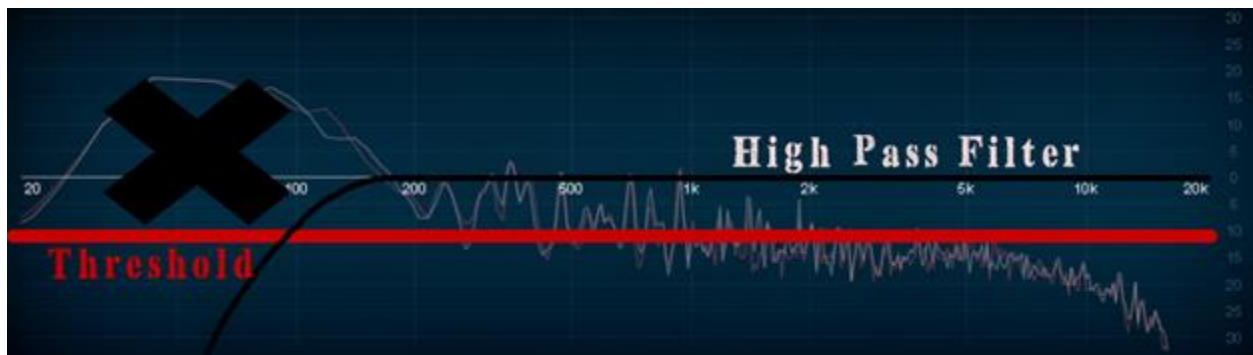


Ilustración 5, Eliminando las componentes carentes de información (zona marcada con una X) mediante el filtro paso altos, o high pass filter, podemos comprimir las fundamentales.

Si no aplicásemos el HPF a la señal antes de comprimirla como en la ilustración 4, estaríamos sobre comprimiendo las componentes graves de la señal, que no contienen información útil de la voz, mientras que el compresor prácticamente no actuaría sobre las frecuencias útiles, situadas de 120Hz hasta los 3KHz, aproximadamente, cuyo nivel promedio nos interesaría igualar para conseguir una señal más uniforme e inteligible. En la ilustración 5 aplicamos un filtro paso altos a la situación de la ilustración anterior, consiguiendo el efecto deseado: elimina las componentes que activaban el compresor de forma ineficaz.

Podemos entonces afirmar a través de lo expuesto, que lo ideal para el cumplimiento del objetivo de este trabajo mediante la configuración de las herramientas expuestas, sería:

1. Que nuestro compresor actuase mediante la configuración del parámetro Umbral (*threshold*) sobre el nivel promedio de la señal de entrada.

2. Que fuéramos capaces de conseguir que las frecuencias que llegasen a este umbral de compresión fueran las fundamentales de la voz y no otras (graves, sin información) provenientes de ruidos o interferencias de cualquier tipo, lo que vamos a conseguir, parcialmente, filtrando la señal con un high pass filter, o filtro paso altos, que no es más que un ecualizador configurado de forma que elimina las componentes graves de la señal por debajo de determinada frecuencia de corte.
3. Que con las funcionalidades descritas consiguiésemos tener una interfaz fácil de usar, medianamente auto configurable y que cumpla de la forma más sencilla posible con las expectativas del usuario.
4. Que pudiéramos estudiar la mejora en la inteligibilidad de la señal procesada respecto a la misma señal sin procesar de la forma más objetiva posible.

Vamos a desarrollar, bajo el standard VST y el framework de JUCE, un plugin que será compatible con cualquier plataforma de streaming, como las antes citadas, en su versión standalone, así como la versión VST que nos permitirá insertarlo en OBS.

JUCE es un framework de programación multiplataforma, reconocido por la alta calidad de su API gráfica y por ofrecer opciones de exportación avanzadas adaptándose a los citados estándares, entre ellos VST, en función de la plataforma para la que lo decidamos compilar.

Capítulo 2 - Componentes y métricas de una señal de voz y su relación con la compresión

FICHA TEMÁTICA

14.1

Compresión y limitación

Un compresor es un dispositivo capaz de hacer que el nivel de la señal de salida varíe a un ritmo diferente a como lo hace la señal de entrada. Por ejemplo, un compresor con una relación de 2:1 dará un nivel de salida que cambia sólo la mitad de lo que lo hace la entrada, a partir de un cierto umbral (ver figura). Así pues, si el nivel de entrada fuera a cambiar en 6 dB, la salida sólo variaría en 3 dB. Existen también otras relaciones de compresión tales como 3:1, 5:1, etc. Con las relaciones más altas, el nivel de salida varía sólo en una pequeña cantidad respecto a los cambios en el nivel de entrada, lo que hace útil a este dispositivo para reducir el margen dinámico de una señal. El umbral de un compresor determina el nivel de señal por

encima del cual tendrá lugar la acción.

Un limitador es un compresor con una relación de compresión muy alta. Se utiliza para asegurar que el nivel de la señal no se eleve por encima de un determinado umbral. Un limitador «suave» tiene una acción que funciona sólo suavemente por encima del umbral, en lugar de actuar como si fuera un muro, mientras que un limitador «duro» tiene casi el efecto de recortar cualquier señal que exceda el umbral.

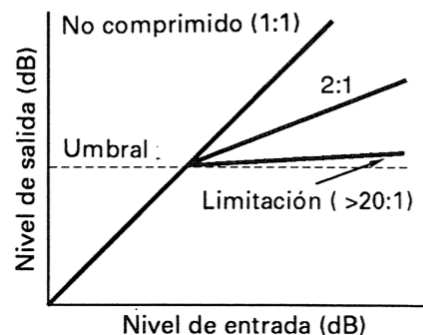


Ilustración 6, Ficha temática dedicada a la compresión (McCormick, 1996)

2.1 Introducción a la compresión de rango dinámico

Hoy en día es abrumadora la cantidad de marcas comerciales que ofrecen micrófonos con distinto tipo de respuesta en frecuencia y diagrama polar² de captación, y el abanico de calidades diferentes que se abre: es hardware que crea señales distintas, no estandarizadas, que han de ser tratadas según su naturaleza para optimizar y unificar, en el caso de una señal vocal, su rango dinámico y ecualización, para mejorar su calidad y comprensión. Esta variedad explica en cierta manera la complejidad de estos procesadores y la

² Gráfica de dos dimensiones que expresa la caída en decibelios en la captación según el ángulo de incidencia de la onda sonora al micrófono, de 0 a 360°.

amplia gama de aplicaciones que ofrecen, aunque en su origen eran aparatos electrónicos no demasiado complejos: constaban de una serie de válvulas o tubos de vacío, y sin más controles que una entrada y una salida, ofrecían un control simple, que se fue haciendo más complejo con la evolución tecnológica en décadas posteriores. Con la evolución del audio analógico y digital profesional, compañías privadas como Waves³ y Steinberg, han desarrollado sus propios algoritmos optimizados que “copian” el funcionamiento de equipos analógicos de alta gama, pero esta increíble evolución, lleva un crecimiento en la complejidad de la estructura del reflejo software de estos avanzados equipos hardware. Tiempo de ataque, tiempo de liberación, umbral de actuación, ratio de compresión y ganancia de salida son algunos de los parámetros de los compresores más simples hoy en día. Hasta un usuario avanzado y con facilidad de adaptación, tendría que buscar para qué sirven estos parámetros y cómo afectan a la señal y a su sonoridad, antes de pararse a configurarlos, teniendo que pasar por un largo periodo de experimentación y entrenamiento auditivo para comprender su funcionamiento.



Ilustración 7, Compresor comercial WAVES software, imitación de una reconocida unidad Analógica de alta gama, el API-2500 de SSL.

³ En la ilustración 7 vemos uno de sus plugins más famosos de Waves, el compresor API 2500 en su versión software que emula el funcionamiento del original analógico, diseño propietario de la marca Solid State Logic - SSL.

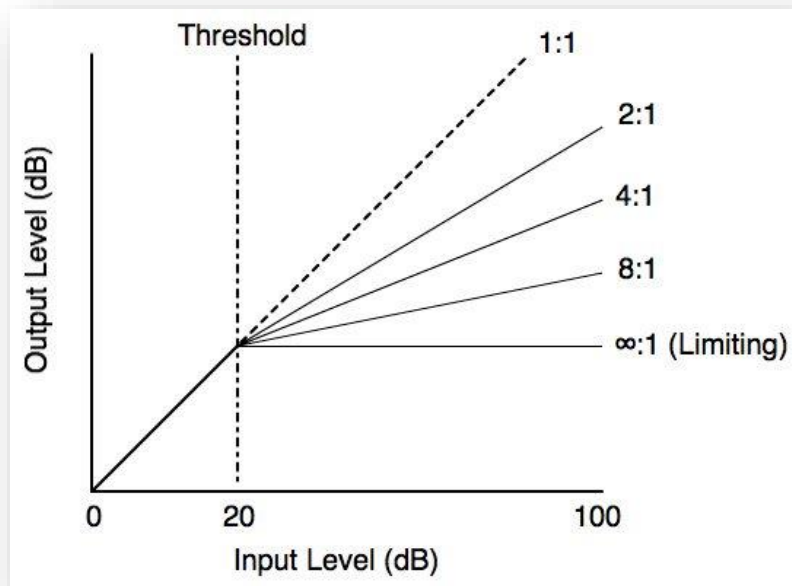


Ilustración 8, Función de Transferencia de un compresor

Durante el procesamiento son comprimidas aquellas partes de la señal que llegan al umbral o lo sobrepasan. La cantidad de compresión aplicada se expresa con el Ratio, por ejemplo 2:1, que indica que por cada 2 Db's que pase del umbral la señal, sólo dejaremos salir 1 de ellos, reduciendo su amplitud de esta forma. En la ilustración 8 vemos la función de transferencia de un compresor de forma gráfica. La pendiente que hasta el punto del umbral es de 45° (relación 1 entrada – 1 salida) disminuye proporcionalmente al ratio a partir de este punto.

- El **Threshold o Umbral** del compresor es el nivel de amplitud en dB's a partir del que empieza a comprimir. Cuando la señal que transita el compresor llega al nivel del Umbral se ve afectada y comprimida según los parámetros del procesador.
- El **Ratio** expresado en forma X:1, expresa el grado de compresión de la señal. La relación expresa que en los puntos donde se pase X db's del umbral, salga únicamente 1 db. Por ello 8:1 es mucho más comprimido que 2:1.

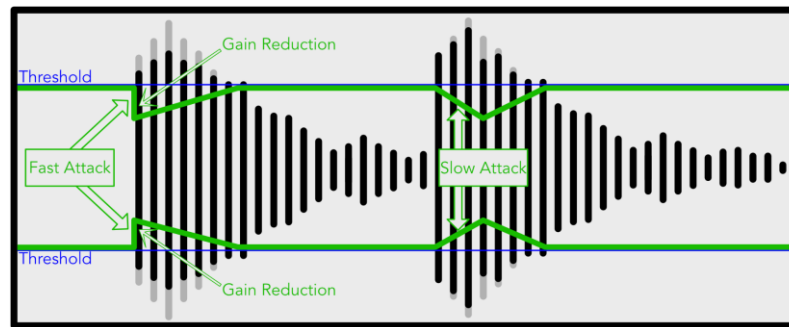


Ilustración 9, Cómo afecta el ataque de un compresor

- El **Tiempo de ataque** o simplemente **Ataque** del compresor, medido en milisegundos, es el tiempo que tarda en completar el total de la reducción de ganancia. Un ataque rápido provocará cambios más bruscos en la señal, mientras que uno lento permitirá cambios suaves en la señal (ver ilustración 9).

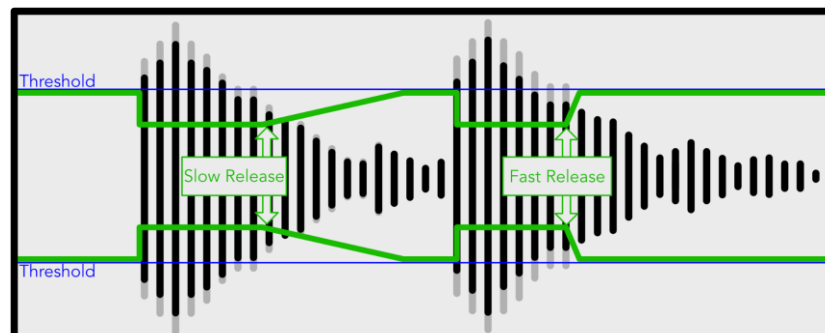


Ilustración 10, Cómo afecta el release de un compresor

- De forma análoga, el **Release**⁴ ó **Tiempo de liberación**, es el tiempo que tarda el compresor en volver del estado de máxima compresión al nivel 0db's de compresión (ver ilustración 10).

⁴ Imágenes de ataque y release tomadas de <https://www.masteringthemix.com/blogs/learn/the-secret-to-compressor-attack-and-release-time>

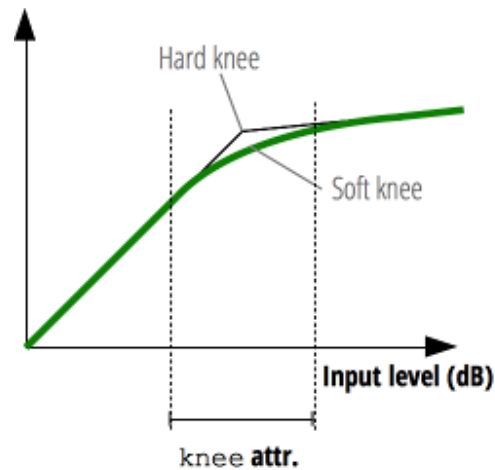


Ilustración 11, Aportación del parámetro Knee a la configuración del compresor

- El parámetro **Knee** consigue que la compresión suene más suave o dura. Ajusta la transición de no compresión a compresión, en el punto del threshold, permitiendo que el efecto de reducción de ganancia sea más suave al **tener forma de curva** o más pronunciado si lo configuramos drásticamente **como una recta**. Es un parámetro que se mueve generalmente entre 0 y 1, siendo 0 totalmente recto o 1 totalmente curvo (ver ilustración 11).

En la ilustración 12 podemos ver cómo la compresión consigue que una señal con una diferencia de amplitud importante entre sus partes altas y bajas termina teniendo un rango dinámico mucho más homogéneo: con menos diferencias entre los picos máximos y mínimo de la señal.

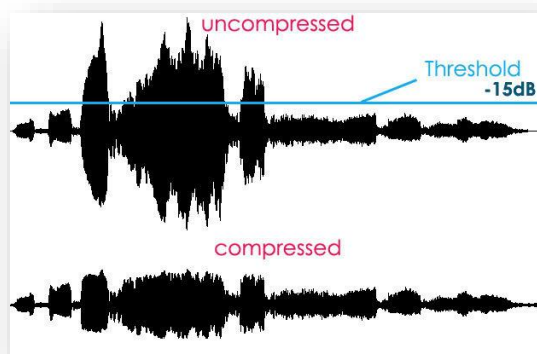


Ilustración 12, Voz sin comprimir vs voz comprimida con un umbral de -15dB:
<http://alternativesilence.blogspot.com/2011/09/un-tutorial-mas-de-compresioncompression.html>

2.2 Componentes frecuenciales audibles de la señal de voz

En la siguiente tabla encontramos, por rangos de frecuencia, la información más importante residente en cada uno. Es importante tener en cuenta las cualidades de la señal previamente a su tratamiento.

Tabla 2, Componentes frecuenciales de la Voz

20 – 80 Hz	La voz no contiene información en este rango frecuencial. Generalmente se corta mediante un filtro paso-alto (<i>high pass filter</i>) para eliminar interferencias, pops de micrófono y reducir el efecto proximidad, que amplifica las frecuencias medias graves en función de la distancia del interlocutor al micrófono.
80 – 300 Hz	En este rango frecuencial suelen residir las frecuencias fundamentales de la voz humana: aquellas que le dan su cuerpo y timbre característico y las de mayor potencia.
300 – 600 Hz	Principalmente armónicos.
600 – 4K Hz	Es donde encontramos las frecuencias responsables de la inteligibilidad de la voz, las "eses" y sus componentes más brillantes.
4K – 8K Hz	De nuevo encontramos armónicos que pueden darle brillo a la señal.
8K– 20KHz	Prácticamente no contiene información. También suele cortarse o filtrarse, en este caso con un <i>low-pass filter</i> o filtro paso bajos, para eliminar sibilancias y ruidos o interferencias agudas en la señal.

En la ilustración 13 se ha incluido una imagen del plugin Waves F6-RTA. Aunque las frecuencias de corte en las componentes de la voz no están estandarizadas y dependen del locutor, en entornos profesionales de audio se

dispone de este tipo de herramientas de precisión para poder modelar la relación frecuencial de la señal. En nuestro desarrollo utilizaremos un filtro paso altos, que sirve al efecto que buscamos, y que eliminará las componentes de la señal por debajo de cierta frecuencia, permitiendo pasar al resto por encima del *cutoff*⁵ escogido.



Ilustración 13, Análisis en tiempo real de voz masculina con headset Plantronics mediante plug-in Steinberg Waves F6-RTA Stereo

La tendencia en el desarrollo de sus herramientas para la mejora de la calidad de la señal emitida, según señalan los roadmaps y estado del arte de Google y Microsoft en sus herramientas de videoconferencias, se está enfocando a la supresión de ruido (Noise Cancellation⁶) principalmente, pasando en todos los casos por sistemas de caja negra mediante redes neuronales. Desde luego queda cubierta la parte del enfoque fácil de usar, pero da lugar a un sistema de transmisión únicamente eficiente para la voz hablada, y como muestra se presenta un ejemplo (Ver ilustración 14) donde la emisión de un instrumento musical, que genera otro tipo de señal, se ve alterada y se

⁵ Frecuencia de corte.

⁶ Disponible para usuarios de Google Workspace desde junio 2020, workspaceupdates.googleblog.com y para usuarios de Microsoft Teams en PC en Noviembre del 2020 mientras que los usuarios de MAC habrán de esperar a Mayo del 2021: <https://www.microsoft.com/es-es/microsoft-365/roadmap?filters=&searchterms=teams%2Cnoise%2C>

aprecia claramente tanto gráfica como acústicamente una degeneración, haciendo inservible el sistema para transmisión de música.

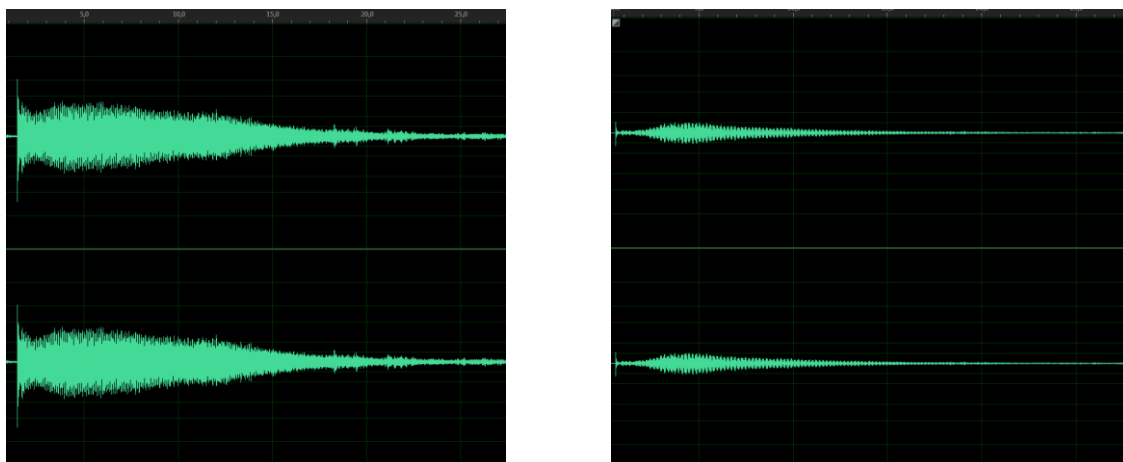


Ilustración 14, Señal de un instrumento desnaturalizada por el reductor de ruido automático de Microsoft Teams

Teams ofrece su servicio de mejoras de audio de forma no configurable. En el ejemplo anterior se aprecia la compresión aplicada, debido al algoritmo de supresión de ruidos y sus mejoras, proporcionando una experiencia pobre en el ámbito musical por estas características automáticas sin posibilidad de configurar más parámetros por parte del usuario.

A nivel académico, las líneas de investigación actuales, a la par que esforzarse en el entrenamiento de las redes neuronales que operan en los sistemas anteriores, van dirigidas al IoT⁷, optimización en la transmisión y pérdida de paquetes durante el streaming⁸ en las redes locales y en general en la capa de red, habiendo encontrado muy pocos estudios y proyectos que hayan tratado de conciliar el tratamiento analógico de la señal en su emisión con los

⁷ Improving audio streaming over multi-hop ZigBee networks, DOI: 10.1109/ISCC.2008.4625771

⁸ A survey of packet loss recovery techniques for streaming audio, DOI: 10.1109/65.730750

modernos sistemas de IA y la mejora de audio en las plataformas de streaming, como se pretende abordar en este Trabajo.

A continuación, presentamos una serie de plug-ins con un objetivo similar al que hemos desarrollado. En la sección 3.1 vemos los incorporados en la principal herramienta de emisión software, OBS, y en la sección siguiente otros que son ofrecidos por Reaper, programa propietario muy conocido por su alta calidad, que permite su uso en modo licencia de evaluación, y provee de una suite de plugins gratuitos muy interesantes, versátiles y configurables, como alternativa a los que trae de serie OBS, también compatibles con el standard VST.

2.3 Plugis ins incorporados en OBS

2.3.1 Compresor de rango dinámico

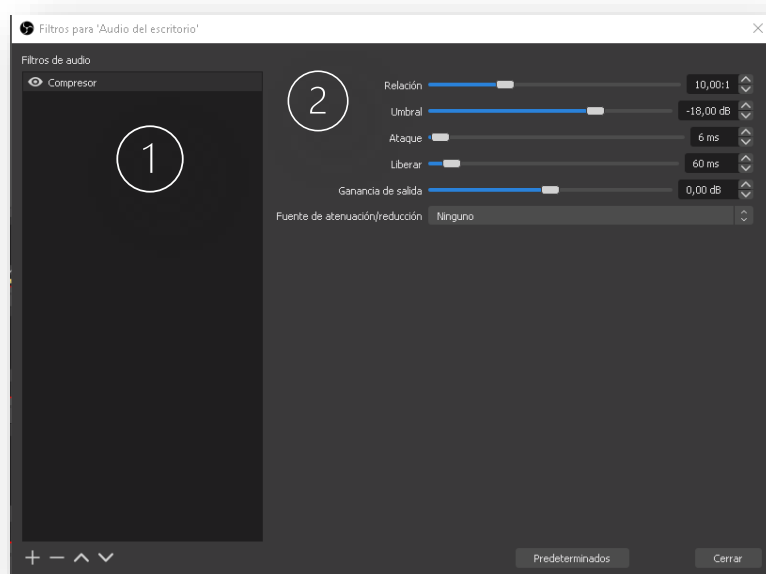


Ilustración 15, Interfaz del compresor incorporado en OBS studio, LTS.

En la ilustración 15 vemos el compresor incorporado en OBS. El área (1) corresponde a la lista de inserciones del canal, en este caso Audio del escritorio. Si se colocasen más inserciones aparecerían en esta lista y se aplicarían a la señal en serie en el orden establecido, accediendo a los parámetros de configuración de cada procesador mediante (2).

En (2) Se encuentra el interfaz de los parámetros del compresor. En este caso han añadido una fuente de *sidechain* al compresor, que hace que actúe como disparador de este, en lugar de la llegada de la amplitud de la señal al umbral, un canal externo, configurable. Por ejemplo, esta opción es útil para configurar un compresor que estará activo durante una emisión de radio, bajando la música en los momentos en los que hable el locutor, cuya señal de voz conectada al *sidechain*, actuará como disparador del compresor, bajando el volumen de la música en un tiempo correspondiente al tiempo de ataque del compresor, y reestableciendo el volumen de la música en un tiempo igual al *release*.

Tras probarlo podemos afirmar que cumple su papel, pero desde luego no es fácil de usar, teniéndose que invertir un tiempo considerable en buscar información sobre cada una de las constantes de tiempo del procesador si no se las conoce, incluyendo el inconveniente de no informar en ningún momento al usuario del nivel de compresión en db's de la señal tiempo real, siendo desconocida esta métrica tan importante en la compresión en entornos técnicos, para saber "qué estamos haciendo".

2.3.2 Limitador

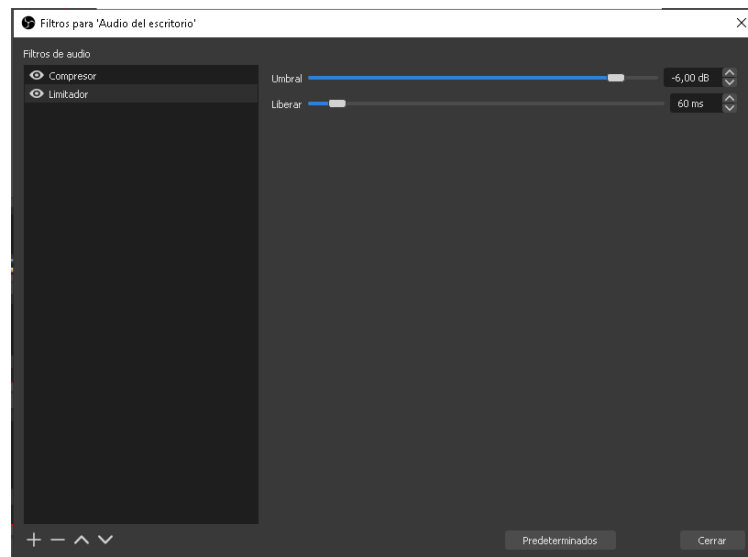


Ilustración 16, Interfaz del compresor incorporado en OBS studio, LTS.

Útil, pero menos versátil que un compresor, un limitador es un compresor con el ratio configurado a un valor infinito, y el ataque a 0 para que actúe inmediatamente. Su función es impedir que la señal sobrepase determinado umbral, para impedir, generalmente, que saturé. En este caso son configurables el Umbral y el tiempo de liberación (Liberar en la ilustración 16), que, correspondiéndose con el release, es el tiempo que la reducción de ganancia tarda en volver a 0 (en el caso ideal de que otro pico no ataque el umbral antes de que eso suceda).

No incluye ningún tipo de plugin de ecualización, filtros paso altos o similares. Sin embargo, contiene un plugin eliminador de ruido con un único parámetro "nivel de eliminación de ruido (0-100)" que hace lo que puede, mediante el mismo algoritmo, sin permitir particularizar, en el contexto de emisiones descrito y su diversidad de señales.

2.4 Plug ins de Reaper

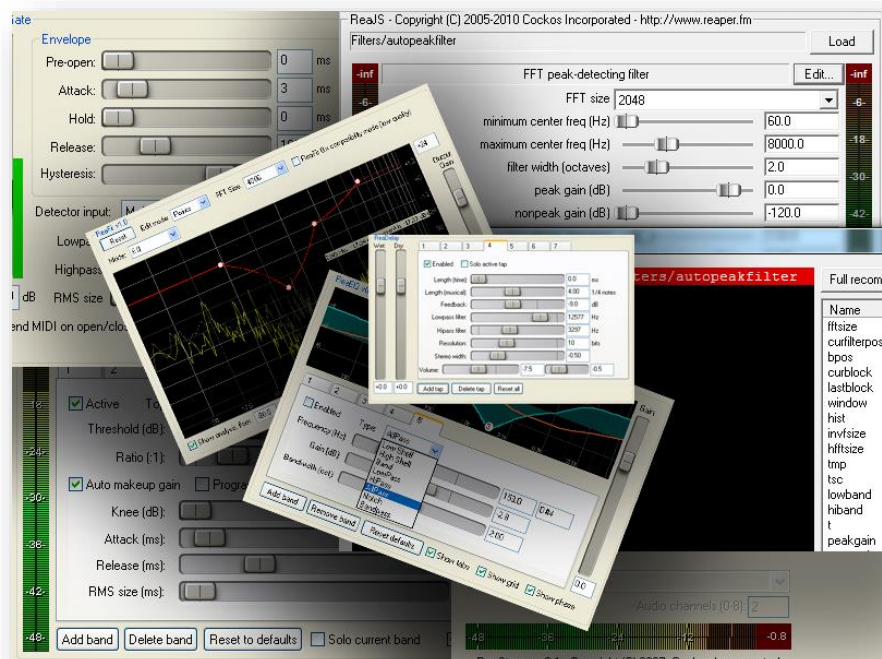


Ilustración 17, El complicado abanico de plugins que ofrece Reaper.

Reaper es un programa de mezcla de audio multipista compatible con VST. Pese a ser propietario, ha ganado muchos adeptos por su gran rendimiento, alta compatibilidad y buena presentación visual. Permiten que se utilice con licencia de prueba, y la suite de plugins base que acompaña al programa es gratuita.

En la ilustración 17 se quiere mostrar de un vistazo la complejidad de los plugins de Reaper, suite cuyas especificaciones técnicas son altamente prometedoras:

Soporte multiplataforma, interfaz de rápida actualización y precisión en la medida, bajo uso de CPU & ram y altamente configurables. Promete como una suite profesional de plugins, y es versátil como tal. Aunque realmente cumplan todas, la descripción de sus funcionalidades queda fuera del ámbito de este trabajo; lo que se quiere mostrar es que estamos ante una suite muy completa de plugins de alto rendimiento multi plataforma extremadamente compleja de manejar que para cualquier usuario con una mente alejada de la ingeniería serían muy difíciles de configurar. No obstante, para el usuario medio-

avanzado destacamos un compresor y un ecualizador: el ReaEQ y ReaCOMP en las ilustraciones 18 y 19 respectivamente.

- ReaEQ

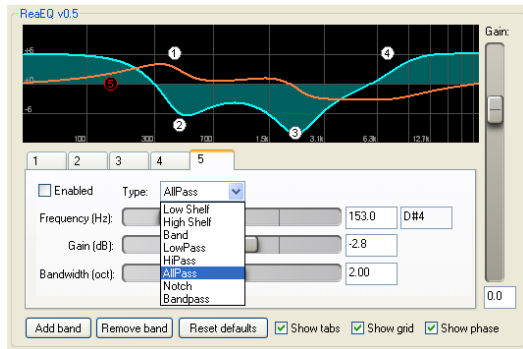


Ilustración 18, ReaEQ, de Reaper

- ReaCOMP

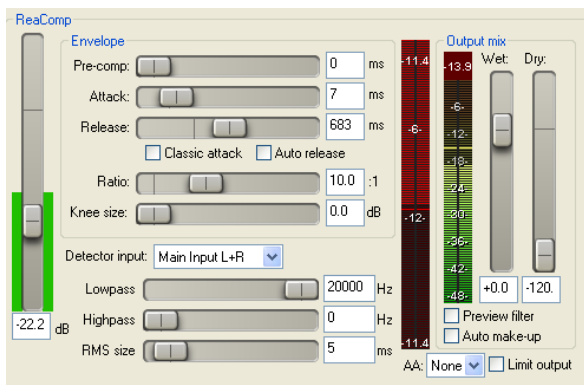


Ilustración 19, ReaCOMP, de Reaper

Ecualizador ultra configurable, con el que se pueden implementar una gran cantidad de tipos de filtros, con posibilidad de habilitarlo por secciones, y una interfaz gráfica muy completa.

Se trata de un compresor altamente configurable con parámetro knee, sidechain, configuración de release automático, y muchas otras opciones que hacen de él una herramienta muy versátil para usuarios con conocimientos técnicos de sonido.

Capítulo 3 - La implementación en JUCE

Juce es un framework de desarrollo diseñado específicamente para producir aplicaciones de audio. De código abierto y multiplataforma, está pensado para que la experiencia del programador sea idéntica a través de distintos sistemas operativos en el uso de la herramienta, y permite generar soluciones tanto para versiones móviles como de escritorio. Compatible con los estándares de desarrollo de plug-ins VST, AU⁹ y AAX¹⁰ comúnmente utilizados en los DAW's industriales, nos permite adaptarnos al entorno mediante esta selección. En este caso se exportará como VST para OBS. Como OBS es compatible con el standard VST, elegimos éste como formato final de nuestro procesador. En la ilustración 20 vemos la ventana de "Nuevo Proyecto" de Juce.

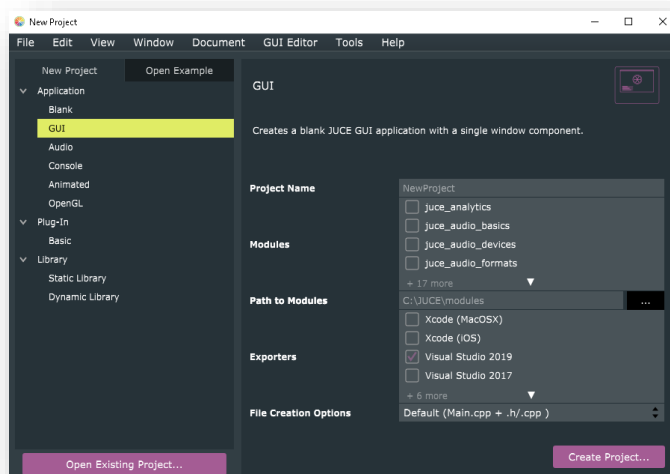


Ilustración 20, Ventana de "nuevo proyecto" en Juce al iniciar el programa.

⁹ Audio Unit, estándar implementado por Core Audio para el desarrollo de plugins en sistemas Apple MacOS e iOS.

¹⁰ Avid Audio eXtension, tecnología análoga a las anteriores pero para el software multi-pista profesional de Avid, Pro-Tools.

Cuando creamos un plug-in desde cero, Juce genera automáticamente 2 clases con sus dos correspondientes cabeceras, el *Procesador* y el *Editor* (ilustración 21). Una vez Juce ha generado el proyecto y el código según nuestras especificaciones, estamos preparados para abrir el proyecto en el IDE escogido y trabajar sobre él, editando el código y las funciones características generadas según el tipo de proyecto.

En este modelo de dos capas, el procesador es la clase donde se encuentra el bucle de procesamiento en tiempo real del buffer de audio, mientras que en el editor configuramos la interfaz, con los controles y parámetros que harán accesible al usuario las configuraciones del plugin y los medidores necesarios para monitorizar el proceso.

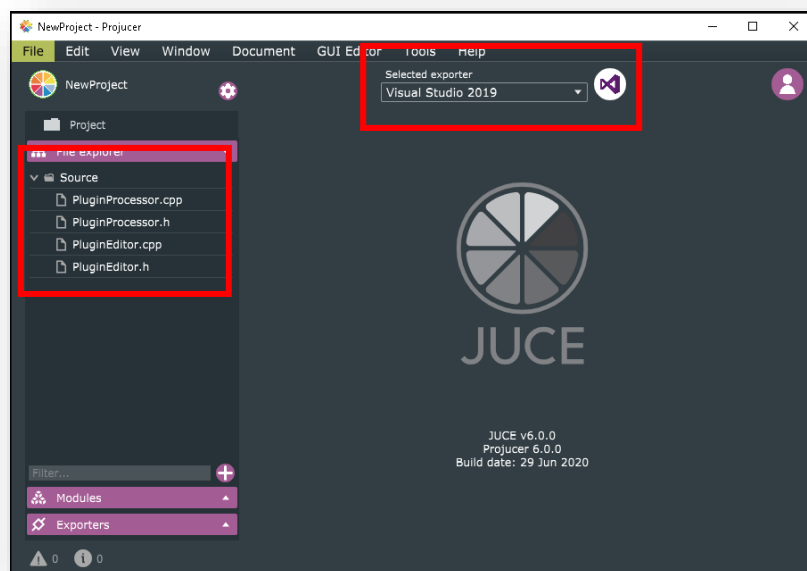


Ilustración 21, Ventana de proyecto recién creado listo para abrirse en Visual Studio 2019. A la izquierda están los archivos generados por Juce al elegir la creación de un audio plug-in. En la parte superior está el selector de IDE que usaremos para editarlos y compilar el proyecto.

El Procesador

El procesador es donde ocurre todo el procesamiento en tiempo real. Está compuesto por la clase `PluginProcessor.cpp` y su cabecera. Al tratarse de un entorno tan versátil hay partes del código generado que no se utilizarán, pues se usan en otros contextos. Sin embargo, en otras partes del código habremos de incidir en función de

lo que queramos implementar y sus implicaciones. Se describen las partes principales del código para aclarar un poco tanto el abordaje general de la elaboración de un plugin como las decisiones tomadas durante este TFG. La documentación oficial de Juce incluye en inglés suficientes ejemplos y exposiciones para el lector interesado y se puede encontrar en <https://docs.juce.com>.

El constructor de la clase instancia el objeto al crear el plugin. En nuestro caso hemos añadido los parámetros necesarios para gestionar la interfaz de usuario al vector *parameters*, tras instanciarlo con un ID único y un nuevo *layout*. La creación del vector *parámetros* nos permite una comunicación directa entre el Editor y el procesador mediante el método *getValue()* de la estructura, que recoge el valor de un parámetro del vector citado, y nos sirve tanto para consultarlo como para actualizarlo o almacenarlo. En el constructor añadimos los *Listeners* necesarios a los parámetros para tener el procesador actualizado respecto al editor en todo momento, mientras se interactúa con los controles deslizantes en la vista del programa. En la ilustración 22 vemos el constructor y el destructor de la clase *AudioProcessor*.

```
SimpleCompressorAudioProcessor::SimpleCompressorAudioProcessor() :
{
    #ifndef JUCE_PLUGIN_PREFERRED_CHANNEL_CONFIGURATIONS
        AudioProcessor(BusesProperties())
    #endif
    #if ! JUCE_PLUGIN_IS_MIDI_EFFECT
    #if ! JUCE_PLUGIN_IS_SYNTH
        .withInput("Input", AudioChannelSet::stereo(), true)
    #endif
        .withOutput("Output", AudioChannelSet::stereo(), true)
    #endif
    },
    #endif
    parameters(*this, nullptr, Identifier("SimpleCompressorState"), createParameterLayout())
{
    parameters.addParameterListener("threshold", this);
    parameters.addParameterListener("knee", this);
    parameters.addParameterListener("attack", this);
    parameters.addParameterListener("release", this);
    parameters.addParameterListener("ratio", this);
    parameters.addParameterListener("makeUp", this);

    parameters.addParameterListener("bypass", this);
    parameters.addParameterListener("hpf", this);

    parameters.addParameterListener("frequency", this);
    parameters.addParameterListener("quality", this);
    parameters.addParameterListener("compDegree", this);
    parameters.addParameterListener("autoThreshold", this);
}

SimpleCompressorAudioProcessor::~SimpleCompressorAudioProcessor()
{
}
```

Ilustración 22, Constructor y destructor del Procesador de Audio.

El destructor, libera de forma segura la instancia. Por diseño, antes de cerrar un plugin de Juce, se llama a la función `releaseResources()` de la ilustración 23, que destruye los punteros aún vivos y libera los recursos convenientes para que la ejecución pueda terminar correctamente. En nuestro caso liberamos los punteros creados para gestionar el ecualizador, y llamamos a la función `reset()` que reinicia el estado interno del compresor.

```
void SimpleCompressorAudioProcessor::releaseResources()
{
    if (outputs) {
        free(outputs);
        outputs = nullptr;
    }
    if (pole1) {
        free(pole1);
        pole1 = nullptr;
    }
    if (pole2) {
        free(pole2);
        pole2 = nullptr;
    }
    if (pole3) {
        free(pole3);
        pole3 = nullptr;
    }
    if (pole4) {
        free(pole4);
        pole4 = nullptr;
    }
    compressor.reset();
}
```

Ilustración 23, Liberación de los punteros utilizados por el Filtro paso Altos y reseteo del Compresor para liberar los recursos asociados a ambas estructuras.

La función `processBlock` es donde ocurre todo el procesamiento de audio en tiempo real a través del buffer. Para aclarar su funcionamiento se cita un ejemplo en la ilustración 24, presente en la documentación de Juce, en el que en cada instante se aplica una ganancia, cuyo valor se lee del interfaz del plugin y puede modificarse mediante un slider, en tiempo real, afectando a la ganancia de la señal.

```
void processBlock (juce::AudioSampleBuffer& buffer, juce::MidiBuffer&) override
{
    buffer.applyGain (*gain);
}
```

Ilustración 24, Función de proceso en tiempo real del buffer de Audio.

La función `prepareToPlay()` de la ilustración 25 es llamada cada vez que vamos a reproducir audio a través del plugin, y antes de que éste sea procesado. En nuestro caso basta con inicializar el compresor y reservar espacio para los punteros y las variables del ecualizador.

```
void SimpleCompressorAudioProcessor::prepareToPlay(double sampleRate, int samplesPerBlock)
{
    compressor.prepare({ sampleRate, static_cast<uint32> (samplesPerBlock), 2 });

    twopi = 8.0f * atan(1.0f);
    quality = 0;
    pole1 = 0;
    pole2 = 0;
    pole3 = 0;
    pole4 = 0;
    outputs = (float*)calloc(getTotalNumInputChannels(), sizeof(float));
    pole1 = (float*)calloc(getTotalNumInputChannels(), sizeof(float));
    pole2 = (float*)calloc(getTotalNumInputChannels(), sizeof(float));
    pole3 = (float*)calloc(getTotalNumInputChannels(), sizeof(float));
    pole4 = (float*)calloc(getTotalNumInputChannels(), sizeof(float));
}
```

Ilustración 25, Función `prepareToPlay`, que reserva los recursos utilizados en `processBlock`.

El Editor de Audio

Crea la interfaz del procesador y permite al usuario interactuar con los parámetros. El constructor es lo primero que se llama cuando el plugin se instancia, y únicamente se llama en esta inicialización. En este caso se le pasa el procesador, se construye el vector de parámetros y la vista del programa, añadiendo los controles del procesador. En JUCE, la clase `SliderAttachment` permite mantener la conexión entre el slider del editor y el parámetro del procesador.

En la función paint definimos el aspecto: color, texto, botones... sus posiciones y características, mientras que en resized gestionamos las coordenadas de los componentes hijos, como los botones de baipás, high pass filter, o el de cambio de preset, así como los sliders que controlan los parámetros del compresor.

Manteniendo el aspecto del plugin "simple compressor" (Rudrich, 2019) en el que está basado el diseño, se han añadido 3 atributos booleanos que permiten establecer las opciones de bypass, high pass filter y autothreshold, y un slider configurado como un contador de 1 a 5 con el que se gestiona la selección del preset. En la ilustración 26 se ve el aspecto final del plugin bautizado **Improving streaming** desarrollado durante este trabajo de fin de grado.



Ilustración 26, Aspecto final del plugin Improving Streaming desarrollado durante este TFG

3.1 El compresor de rango dinámico

Se ha utilizado el proyecto GNU de Daniel Rudrich de 2019 (Rudrich, 2019) como esqueleto del proyecto. Como figura en la documentación de Git-Hub él utiliza la clase de visualización de la suite de Plug-Ins IEM¹¹ como prueba de implementación de un mecanismo de exploración del buffer de forma anticipada llamado LookAhead. El Look-Ahead es una característica que explora el buffer durante el cálculo de la compresión con cierta anticipación (5-10ms) para ofrecer cálculos más exactos y suaves de la reducción de ganancia ofreciendo una respuesta más cercana a los compresores analógicos. Originalmente su proyecto estaba destinado a estudiar esto, y algunas características del sidechain de los compresores. En nuestra aplicación no tenía sentido la utilización del Look-Ahead por estar orientado a la transmisión en tiempo real y hemos aprovechado la parte de la compresión para implementarla como tal, tras el filtro paso altos y la auto-detección de RMS en el buffer. En la ilustración 27 se ve el aspecto del plugin original.

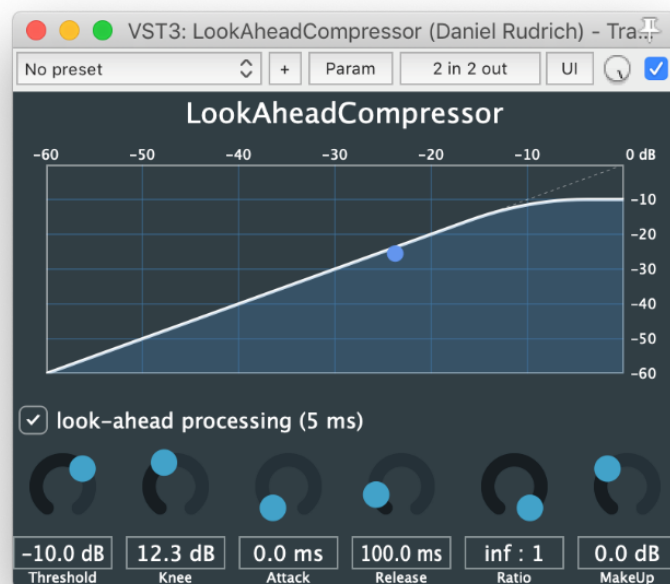


Ilustración 27, Compresor original de Daniel Rudrich, 2019, GNU, Git-Hub available.

¹¹ Suite de plugins gratis y de código abierto desarrollada por el instituto de Música electrónica y Acústica de Graz, Austria (<https://plugins.iem.at/>)

```

void SimpleCompressorAudioProcessor::processBlock (AudioBuffer<float>& buffer, MidiBuffer& midiMessages)
{
    ScopedNoDenormals noDenormals;
    auto totalNumInputChannels = getTotalNumInputChannels();
    auto totalNumOutputChannels = getTotalNumOutputChannels();

    const int numSamples = buffer.getNumSamples();

    // clear not needed output channels
    for (auto i = totalNumInputChannels; i < totalNumOutputChannels; ++i)
        buffer.clear (i, 0, numSamples);

    AudioBlock<float> ab (buffer);
    ProcessContextReplacing<float> context (ab);
    compressor.process (context);
}

```

Ilustración 28, función original processBlock de Daniel Rudrich

En la ilustración 28 se ve en código cómo crear un objeto *AudioBlock* a partir del buffer y procesarlo. En este trabajo se ha aplicado la compresión sobre el buffer después del filtro paso-altos, que paso a describir ahora.

3.2 Filtro Paso Altos

Su implementación se ha basado en el código GPL de *alermi*¹² que implementa un filtro paso altos de tipo IIR¹³. Podemos implementar dos tipos de filtros para audio digital: de respuesta finita e infinita, más conocidos como FIR e IIR respectivamente. Mientras que los de tipo FIR usan sólo muestras de la entrada para obtener un valor de la muestra actual, los de tipo IIR utilizan, además, las muestras de la salida (retroalimentación) para el cálculo. El filtro utilizado es configurable en sus dos parámetros, *Quality* y *Frequency* o Calidad y Frecuencia.

¹² <https://github.com/alermi/JUCE-HighPass>

¹³ Infinite Impulse Response o respuesta infinita al impulso.

- **Quality:** Determina la pendiente del filtro, en decibelios por octava¹⁴. Un valor más cercano a 1 supone una pendiente casi vertical desde la frecuencia de corte, mientras que uno más cercano a 0 supone un descenso progresivo, menos brusco, que disminuye la intensidad de las frecuencias graves más paulatinamente. En la ilustración 29 lo vemos gráficamente.

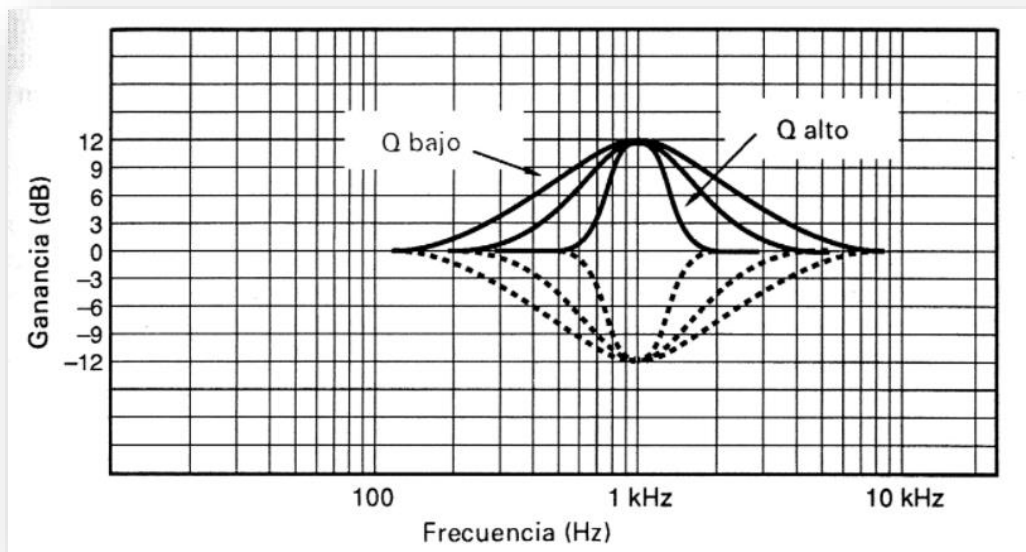


Ilustración 29, Relación entre el parámetro Q y la pendiente del filtro (McCormick, 1996)

- **Frequency:** Frecuencia de corte a partir de la cual queremos dejar pasar las componentes de la señal sin que el filtro actúe sobre ellas.

¹⁴ En sonido el incremento de una octava en la escala musical se corresponde a doblar la frecuencia. 440 - 880Hz es el mismo tono una octava más alta y de 2000-4000 Hz se repite el mismo fenómeno.

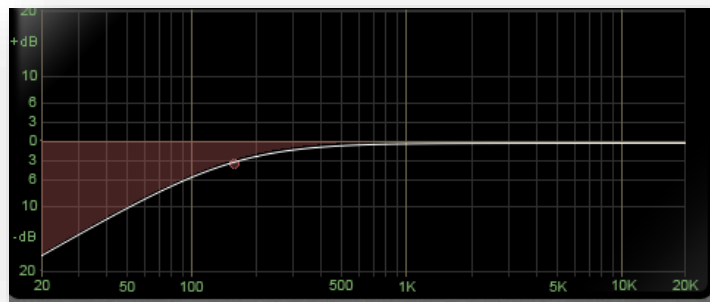


Ilustración 30, Filtro paso altos con una Q cercana a 0: pocos db's/octava

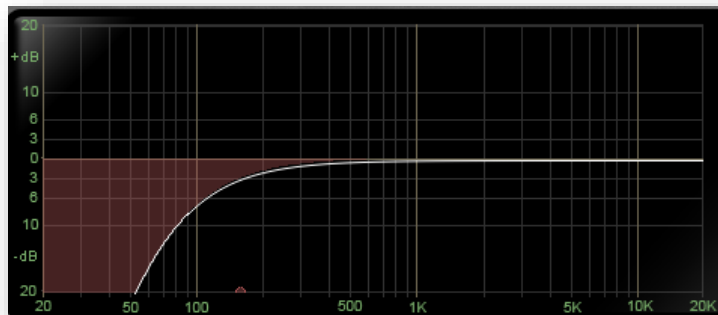


Ilustración 31, Filtro con una Q cercana a 1 con una bajada más brusca

```

frequency = *parameters.getRawParameterValue("Frequency");
float c=(float)exp((-1.0)*twopi*frequency/getSampleRate());
float input;
quality = *parameters.getRawParameterValue("Quality");
for (int channel = 0; channel < totalNumInputChannels; ++channel)
{
    auto* outData = buffer.getWritePointer(channel);
    auto* inData = buffer.getReadPointer(channel);
    for (sample = 0; sample < buffer.getNumSamples(); sample++)
    {
        input = *(inData + sample) - pole4[channel] * quality;

        if (input > 1.0f) input = 1.0f;
        if (input < -1.0f) input = -1.0f;
        input = 1.5f * input - 0.5f * (input * input * input);

        pole1[channel] = input * (1.0f - c) + pole1[channel] * c ;
        pole2[channel] = pole1[channel] * (1.0f - c) + pole2[channel] * c;
        pole3[channel] = pole2[channel] * (1.0f - c) + pole3[channel] * c;
        pole4[channel] = pole3[channel] * (1.0f - c) + pole4[channel] * c;

        //The subtraction causes the filter to do a highpass
        outputs[channel] = input - pole4[channel];

        *(outData + sample) = outputs[channel];
    }
}

```

Ilustración 32, Algoritmo filtro IIR de primer Orden,
por Alermi: <https://github.com/alermi/pd-HighPass>

3.3 La función de Auto-Threshold

Como se dijo en el capítulo dedicado a los objetivos, nos gustaría actuar de forma instantánea en el nivel medio de potencia de la señal. Para su implementación, dentro del bucle de procesamiento en tiempo real *processBlock*, calculamos el RMS del buffer y lo convertimos a Db's calculando el logaritmo en base 10 (ilustración 33).

```

if (autoTh) {
    float tav = 0.5;
    auto rms = 0.0; //1
    rms = (1 - tav) * rms + tav * std::pow(buffer.getRMSLevel(1, 0, buffer.getNumSamples()), 2.0f);
    float dbRMS = 10 * std::log10(rms); //2
    Value thresholdSetTo = parameters.getParameterAsValue("threshold");
    thresholdSetTo = dbRMS;
}

```

Ilustración 33, Código de cálculo del RMS del buffer

Capítulo 4 - Los experimentos

La intención y el reto de esta serie de pruebas es medir de una forma objetiva el impacto del procesador Improving Streaming, en sus diferentes configuraciones, sobre el habla del interlocutor durante una transmisión hecha a través de Twitch.

4.1 Primer experimento: Objetivos y metodología

4.1.1 Objetivos

Los objetivos de este primer experimento han sido,

- a) Probar con determinado nivel de exigencia la herramienta desarrollada y las configuraciones escogidas para los presets (Tabla 4 más adelante), con vistas a desarrollar una versión final calibrando el valor por defecto de los parámetros.
- b) Encontrar un procedimiento para poder medir de forma objetiva el impacto de la ecualización y la dinámica en la inteligibilidad del texto.

Con este fin se escogieron las siguientes métricas a estudiar, utilizadas en ambos experimentos:

- Speech to Text de Google¹⁵ / Microsoft¹⁶
 - Buscando contrastar el número de palabras reconocidas de forma distinta, de ambos servicios: de la señal emitida sin ningún tratamiento, respecto a la misma señal emitida con los presets propuestos.
 - Número de palabras en común: Gcomún y Mcomún, de ambos reconocedores. (ej, Gcomún del preset 1, es el número de palabras en común que tienen, los textos reconocidos por Google para la voz

¹⁵ <https://cloud.google.com/speech-to-text/>

¹⁶ <https://azure.microsoft.com/en-us/services/cognitive-services/text-to-speech/>

emitida sin procesar, respecto a la misma señal emitida con el procesador activado en el preset 1 con el filtro encendido)

Se ha seguido la misma operativa para los servicios Speech to Text de Microsoft y Google utilizando para ello el servicio gratuito de prueba que facilitan a tal efecto cuya URL puede encontrarse en el pie de la página anterior.

Tabla 3, Ejemplo de resultado con el comando *dwdiff*

Text recog. @ bypass	Text recog. @ preset 1	común	añadidas	cambian
Hago matemáticas en verano	Hago prácticas verano	66%	0%	33%

```
kali@kali:~$ echo "Hago matemáticas en verano" > 1.txt
kali@kali:~$ echo "Hago prácticas verano" > 2.txt
kali@kali:~$ dwdiff -siIA best 1.txt 2.txt
Hago [-matemáticas en-] {+prácticas+} verano
Precedente: 4 words  2 50% en común 0 0% suprimidas  2 50% cambiaron
Nuevo: 3 palabras  2 66% en común  0 0% añadidas  1 33% cambiaron
```

Ilustración 34, workflow de comparación entre 1 (texto reconocido sin procesar) y 2 (texto reconocido con el procesador activado en uno de los presets y el filtro paso altos activado)

En la tabla 3 vemos un ejemplo del resultado que arroja el comando *dwdiff* utilizado para comparar la salida de los reconocedores, y en la ilustración 34 el workflow utilizado para las comparaciones.

- Stoi (Crees H. Taal, 2010)

En este trabajo los autores desarrollaron una medida de mejor rendimiento computacional que las habituales (AI, Articulation Index o el STI, Speech Transmission Index (Jianfen Ma, 2009)) métricas de inteligibilidad. Se basa en una medida de inteligibilidad sobre la Transformada de Fourier de la señal dividida en regiones cortas. Los autores proveen de una versión implementada para Matlab con su proyecto, que

hemos usado para realizar las mediciones.

Este script arroja un resultado entre 0 y 1, estando definido así:

$$D = \text{stoi}(\text{sigclean}, \text{sigproc}, fs)$$

Donde *sigclean* y *sigproc* denotan la señal "limpia" y la señal de voz procesada, respectivamente, y *fs* es la frecuencia de muestreo en Hz.

Se espera que la salida *D* tenga una relación proporcional con la inteligibilidad del habla, donde una *d* más alta, más cercana a 1, denota mejor inteligibilidad, mientras que con un resultado más cercano a 0 estamos en el caso contrario, por lo que sacando un 0,5 no habría mejora ni empeoramiento en la percepción de la voz.

Se han realizado las grabaciones de audio con

- Primer experimento: Micrófono incorporado portátil HP año 2014 (calidad media-baja, aparato usado).
- Segundo experimento, el micrófono incorporado de un portátil de gama media-alta comprado recientemente (MSI GL63 8RD)

4.2 Primer experimento

4.2.1 Hipótesis

La compresión y el filtro paso altos pueden ayudar en gran medida a la inteligibilidad de la señal

4.2.2 Método

En ambos casos he tratado de acercarme lo más posible a un equipo de grabación básico, que podamos usar cualquiera durante nuestras videoconferencias, y sea de uso extendido, buscando un entorno realista en las pruebas y no uno ideal.

Se ha emitido utilizando OBS con el plugin *Improving Streaming* como inserción¹⁷ en la salida de OBS a través de TWITCH, primero en bypass y posteriormente con cada uno de los presets configurados y las características de Auto-Threshold y High Pass Filter encendidas, de forma que, al finalizar, hemos podido descargar la emisión del portal de Twitch conteniendo esta, 6 regiones de audio: 1 emitida en bypass, y otras 5 correspondiendo cada una a los 5 presets descritos del 1-5.

Se ha editado la emisión con Adobe Audition y recortado con precisión mediante herramientas de detección de silencio automáticas, para producir archivos de audio ajustados a sus duraciones, de la misma longitud.

Se han cargado los archivos en Matlab (\\DocumentosAnejosMemoria\\Experimentos\\ScriptsConResultadosSTOI) y calculado el STOI mediante el script incluido en [Crees H. Taal, 2010]. En la siguiente tabla figuran las configuraciones del procesador durante la emisión del primer experimento:

Tabla 4, Presets del procesador en el primer experimento

	Attack (ms)	Release (ms)	Ratio (X:1)	Threshold (dB's)	Knee (0-1)	Make Up (dB's)
1	50	15	3	Auto	0.5	3
2	45	25	3.5	Auto	0.5	4
3	40	50	4	Auto	0.5	5
4	25	60	4.5	Auto	0.5	5.5
5	15	80	5	Auto	0.5	6
ideal	20	100	3	-32	0.5	2

¹⁷ Insertado en la salida, procesando esta antes de ser emitida.

La idea de la elección de los tiempos de ataque y liberación es que crezca el grado de compresión a medida que aumenta el número de preset, cerrando el ataque y aumentando el release, respectiva y paulatinamente, a tal efecto.

4.2.3 Resultados

En la tabla 5 vemos los resultados del primer análisis, y en la ilustración 35 la gráfica del STOI y su evolución según el preset, lo que nos da una idea de la mejora en la inteligibilidad según aumenta éste y con él, la compresión.

Tabla 5, Resultados primer experimento

preset	Mcomun	Maniadidas	Mcambian	Gcomun	Ganiadidas	Gcambian	STOI
1	83,000%	0,000%	0,000%	78,000%	3,000%	18,000%	65,915%
2	82,000%	2,000%	1,000%	88,000%	2,000%	9,000%	66,039%
3	97,000%	2,000%	0,000%	87,000%	0,000%	12,000%	66,104%
4	98,000%	1,000%	0,000%	88,000%	0,000%	12,000%	66,262%
5	96,000%	2,000%	1,000%	88,000%	0,000%	11,000%	66,488%

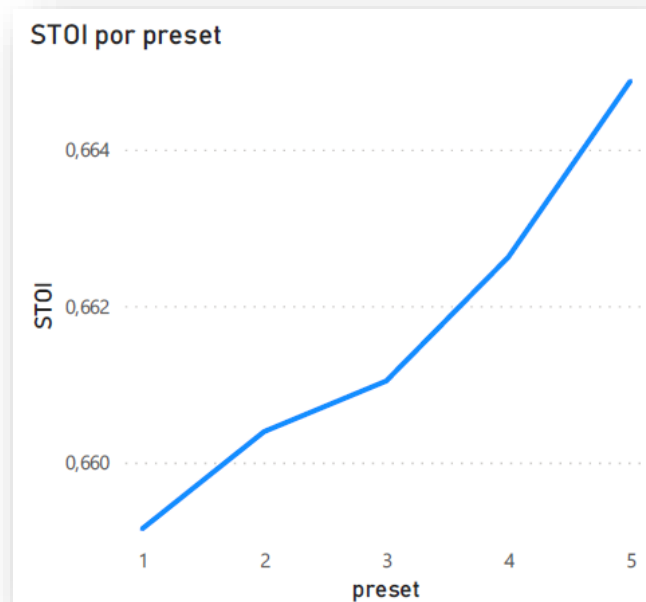


Ilustración 35, Evolución del STOI en cada uno de los presets durante el primer experimento

Podríamos decir que la voz se hace más inteligible, pero en un orden muy pequeño, del orden de décimas estando el resultado del STOI en torno al 66% (ver valor STOI tabla 5).

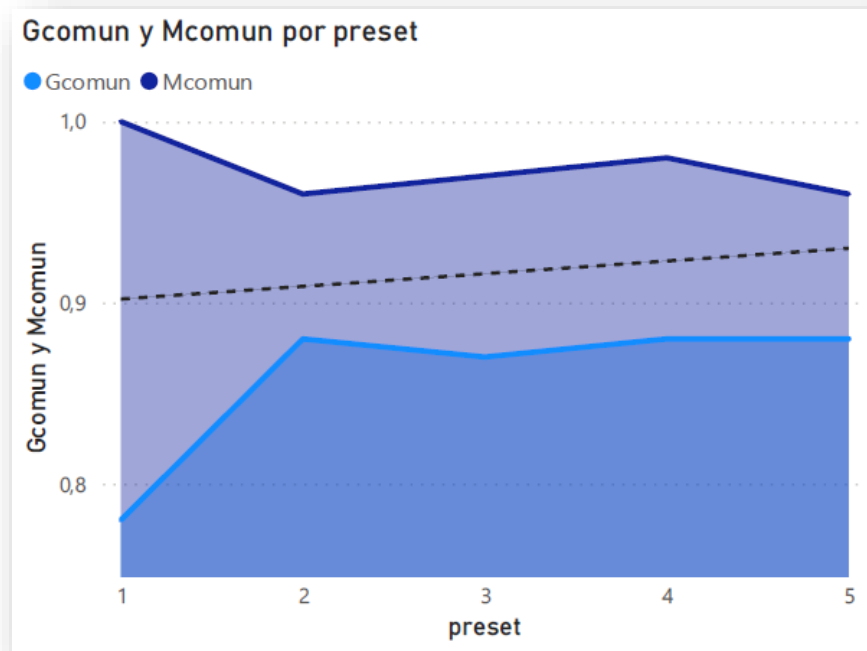


Ilustración 36, Evolución según el preset, de las palabras reconocidas en cada uno respecto a la emisión en bypass, en los reconocedores de Google y Microsoft

En cuanto al número de palabras en común reconocidas, da la impresión de que el valor se estabiliza cuando aumenta el preset (líneas clara y oscura, presets 2, 3, 4 y 5), aunque la pendiente de la recta de tendencia es todavía menor, como vemos en la ilustración 36, y, salvo con el primer preset que corresponde al de menor grado de compresión donde hemos obtenido un resultado más divergente entre los dos reconocedores, con el resto, hemos obtenido un número de palabras reconocidas bastante parecido, sin tener en cuenta la pequeña diferencia de precisión entre ambos servicios de voz a texto.

A primera vista de las gráficas, podríamos decir que mejora la inteligibilidad, que las palabras coincidentes reconocidas tienden a estabilizarse, y que el número de

palabras distintas tiende a ser menor según aumenta la compresión, pero los órdenes de mejora son demasiado pequeños para apresurarnos a concluir nada. Utilizando el análisis inteligente de Power BI (ilustración 37), vemos que detecta como elemento influyente clave el STOI para el aumento del número de palabras en común del reconocedor de Microsoft, pero seguimos manejando órdenes bajísimos de mejora.

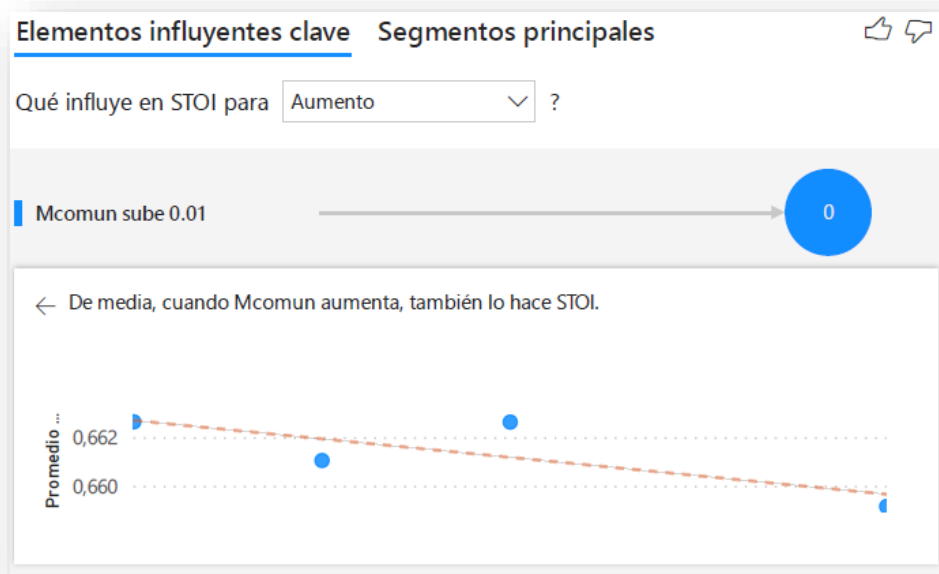


Ilustración 37, Resultados de PowerBI analizando los elementos influyentes clave de los datos de STOI recogidos respecto al número de palabras en común del Speech to Text de Microsoft.

Como el método elaborado es objetivo y los resultados hacen entrever que las hipótesis presentadas quedan demostradas, se han tomado varias decisiones en este punto:

- Aunque tenemos indicios de que ciertamente hay una influencia positiva sobre la inteligibilidad del texto proporcional al grado de compresión, tenemos datos insuficientes para pronunciarnos respecto a la hipótesis, sobre todo debido al bajo orden de mejora medido y se propone extender el

experimento a un segundo, con 4 personas: Hombre Adulto, Mujer Adulta, Hombre Joven y Mujer Joven.

- Para crear una herramienta más versátil que pueda servir a otros efectos (usos de un compresor que requieran valores de release más altos¹⁸), se propone cambiar los presets elegidos, respetando un crecimiento en la compresión: en los presets de 1 a 3 con releases más rápidos, ofreciendo en 4 y 5 un valor más lento, de 800ms de tiempo de liberación, dando más versatilidad a la herramienta, y brindándonos la oportunidad de estudiar si con valores más lentos de este parámetro podemos mejorar la inteligibilidad de la transmisión.
- Para la realización de las siguientes pruebas se eliminará el parámetro auto-threshold para actuar en el umbral lo más optimizado posible para cada interlocutor, y se quitará la configuración "make up" de todos los presets estableciéndola a 0 (sin ganancia) para evitar saturación.
- Vamos a añadir un medidor visual para tener una referencia en tiempo real de la cantidad de decibelios comprimidos y poder ajustar mejor el Umbral durante las pruebas. Este medidor tendrá cuatro estados:
 - Negro: [0 – 2) dbs.
 - Verde: [2 – 4] dbs.
 - Verde-amarillo: (4 – 6] dbs.
 - Amarillo: (6 – 8]
 - Rojo: (8 , Inf)

Estando su estado óptimo fijado en el rango 2-4 db's ya que, una reducción de 3 db's en la amplitud de un sonido equivale matemáticamente al aumento del doble de

¹⁸ Limitación de la señal de audio, emisión de música o música y voz a la vez, aunque al no tener que ver con la vocalización queda fuera del ámbito de este TFG donde será estudiado el impacto de releases largos en la vocalización durante el segundo experimento (presets 4 y 5).

la distancia a la fuente, o lo que es lo mismo, a la percepción del sonido a mitad de su intensidad¹⁹.

4.3 Segundo experimento

Con el fin de generar un procesador en el que se puedan explotar más modos de funcionamiento del compresor, se han utilizado las constantes mostradas en la tabla 6 en la configuración de los presets.

Previo a la grabación de este set de emisiones, se ajustó el threshold para cada interlocutor de forma que el máximo en dB's de la señal comprimida estuviera entre 2-3 dB's en el primer preset y se mantuvo activado, junto al High Pass filter, todos los experimentos excepto en la emisión con el procesador en baipás, modo en que no actúa y que nos sirve de referencia para comparar con la señal tratada por cada preset y poder comparar el número de palabras reconocidas entre la señal procesada y sin procesar.

Tabla 6, Parámetros del segundo experimento

Preset	Attack	Release	Make Up	Ratio
1	70	70	0	3:1
2	40	80	0	3:1
3	10	90	0	3:1
4	50	800	0	3.5:1
5	25	800	0	3.5:1

Fórmula que relaciona dos valores de potencia de señal para hallar su diferencia en decibelios, y cálculo con $P1=2$ y $P2=1$ para hallar su distancia.

$$D_{dB} = 10 \log_{10}(P1 / P2) = 10 \log_{10}(2) = 3,010299.. dB \approx 3 dB$$

Como se comentó la intención de esta elección es proporcionar un procesador de dinámica más versátil, con presets de reléase rápido, como en el experimento anterior, más apropiados para el habla, pero añadiendo opciones con release más lento, cercano al segundo, ya que en el estudio primero no utilizamos valores mayores que 100 ms, y nos gustaría estudiar la inteligibilidad con estas nuevas constantes de tiempo para ver si obtenemos mejoras más consistentes que en el primer experimento sobre la relación entre la inteligibilidad de la señal y la compresión.

4.3.1 Hipótesis

Podemos obtener resultados más claros sobre la influencia de la dinámica en la inteligibilidad de la voz si utilizamos el compresor en más modos de trabajo y estudiamos más interlocutores.

4.3.2 Método

Se mantiene el método del experimento 1º.

4.3.3 Resultados

Para evitar presentar más tablas y unificar los resultados, se presenta el comportamiento del STOI en los últimos análisis según se avanzaba en los presets (pese a las puntualizaciones, se mantiene el convenio de que a mayor preset mayor grado de compresión) mediante el siguiente ideograma; marcamos con un doble más y un doble menos una pendiente pronunciada positiva o negativa y una menos pronunciada o leve con un único signo. En el caso de la mujer joven, el STOI baja más rápidamente que en el hombre adulto, según vamos subiendo el preset, y sube más rápidamente (se escucha mejor a la mujer adulta según se incrementa la compresión) que al hombre joven en cuyo caso encontramos un incremento muy leve en sus valores de STOI, por lo que no podemos concluir nada ante unos resultados tan arbitrarios considerando los ejemplares por separado. En la sección resultados (DocumentosAnejosMemoria\Experimentos\BrutosyAudiosSpeechToTextConResultados\PrimerExperimento) se presentan todas las tablas en PDF. A continuación, pasamos a hablar de los resultados promedio entre los cuatro sujetos de cada métrica escogida.





	Mujer	Hombre
Joven	 --	 +
Adulto	 ++	 -

Ilustración 38, Evolución del STOI por interlocutor, ideograma, segundo experimento.

preset	Mcomun	Maniadidas	Mcambian	Gcomun	Ganiadidas	Gcambian	STOI
1	82,000%	0,000%	17,000%	98,750%	0,500%	0,250%	75,960%
2	83,000%	0,000%	16,000%	98,250%	0,500%	0,500%	66,750%
3	82,500%	0,000%	16,500%	97,750%	0,250%	1,500%	75,366%
4	82,500%	0,000%	16,500%	98,500%	0,250%	0,750%	75,817%
5	81,000%	0,000%	18,250%	98,500%	0,250%	0,750%	75,723%

Tabla 7, resultados promedio de los 4 interlocutores en el segundo experimento.

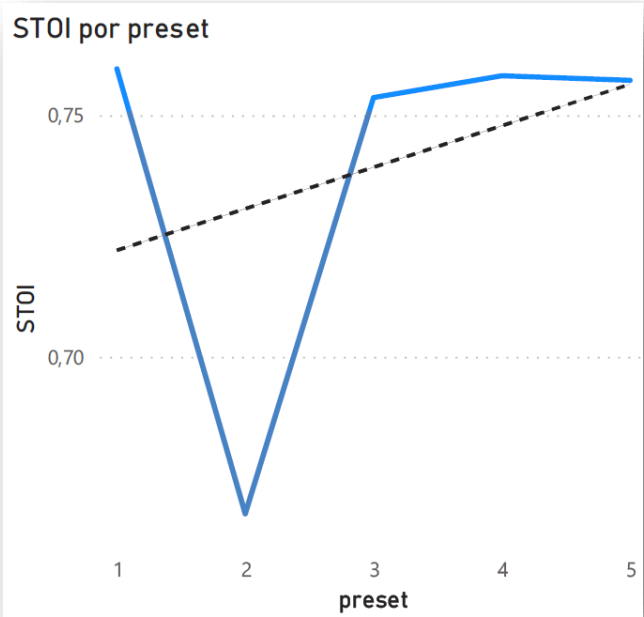


Ilustración 39, STOI promedio por preset, de todos los interlocutores.

Observamos que continua, en promedio, una tendencia al alza del índice de inteligibilidad con una pendiente promedio directamente proporcional al incremento en el preset, y por lo tanto al grado de compresión (ilustración 39).

Efectivamente Power BI detecta como indicador clave la relación entre el STOI y la precisión del reconocedor de Microsoft (ilustración 40). De todas formas, éste último se ha comportado de forma bastante dispar y serían necesarios muchos más ejemplares para poder concluir con el rigor necesario la relación existente o tratar de predecirla.

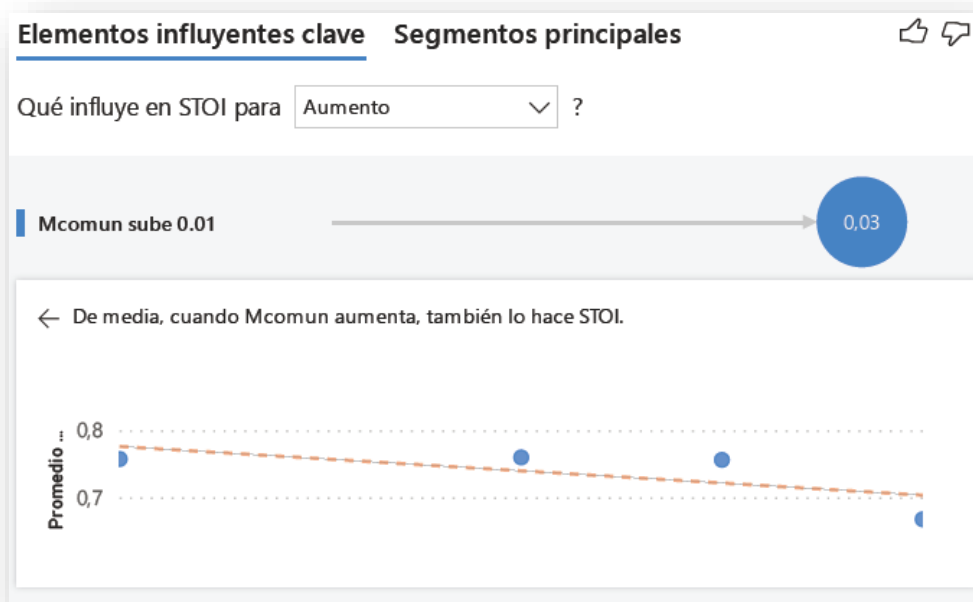


Ilustración 40, El STOI aumenta en este caso ligeramente más que en primer experimento, cuando analizamos los promedios de los 4 ejemplares del segundo. El orden del incremento obtenido sigue siendo muy bajo.

En cuanto al STOI promedio obtenido en el segundo experimento, confirma la hipótesis, y con un orden mayor en la pendiente de la recta de tendencia, aunque los resultados han sido mucho más heterogéneos de lo que se esperaba, así como el comportamiento de los reconocedores pero sobre todo en los valores de STOI obtenidos con cada sujeto. Este hecho es realmente aceptable debido a la gran cantidad de timbres y modos de expresión que tiene la voz humana, y tratar de ajustarnos más,

consistiría en la elaboración de presets específicos, según la expresividad, timbre y tono de cada interlocutor, consiguiendo una mejora consistente en la inteligibilidad mediante una optimización precisa y ajustado a cada caso de las constantes del procesador.

Capítulo 5 - Conclusiones y trabajo futuro

Hemos conseguido entonces desarrollar la herramienta Improving Streaming, utilizando el compresor de (Rudrich, 2019) y el filtro paso altos de (Alermi, 2018). Se han realizado dos experimentos grabando las emisiones a través de Twitch, tratando de ampliar las conclusiones del primero mediante la ejecución del segundo, de la forma descrita en el Capítulo anterior. En ambos experimentos hemos encontrado una relación directa entre el crecimiento del STOI y el de la compresión sobre la señal filtrada por parte de los predictores. Efectivamente los compresores son complejos procesadores de dinámica, porque tienen que tratar señales complejas, y su configuración ha de ser adaptada al tipo de señal que los atraviesa, siendo muy difícil acercarnos al punto óptimo mediante un preset en el caso de la señal hablada, ante la variedad de su tipología. Creemos que este plugin puede aportar y ayudar, como poco, a la mejora en la recepción a un volumen uniforme, de locutores y profesionales que trabajen con programas de emisión software que sean compatibles con la tecnología VST. Aunque no hayamos obtenido unos resultados contundentes por las razones expuestas, la herramienta Improving Streaming funciona muy bien ofreciendo lo que buscábamos: un procesador de dinámica y filtro de señal semi-automático que cualquier persona, sin demasiados conocimientos informáticos, pudiera configurar y llegar a utilizar.

Como conclusión podríamos añadir que hemos aportado algo al mundo GNU, como se trató desde el comienzo del proyecto, ya que queda disponible bajo esta licencia y a partir de ahora en el repositorio de github siguiente: <https://github.com/ealcober/Improving-Streaming.git>. Hemos generado un software auto-configurable, que permite un tratamiento adaptado a cada tipo de voz (salvo el Threshold y el nivel de salida, cuya configuración no se ha automatizado y se ha dejado en manos del usuario), permitiendo aplicar diferentes grados de compresión sin degradar la señal original, a menos que se active el filtro paso altos en el caso de que se quiera transmitir voz hablada con menos "ruido" en frecuencias graves, por lo que consideramos cumplidos los objetivos iniciales del proyecto. Si el usuario no sabe o no

quiere ajustar el threshold también se incluye la opción de automatizar la configuración instantánea de este parámetro con un simple tick.

Nos hubiera gustado también invertir el tiempo necesario para optimizar el rendimiento y el orden de complejidad de la función de procesamiento en tiempo real, pero a efectos del estudio de la inteligibilidad nos centramos en el cumplimiento del objetivo del trabajo, quedando esto fuera del ámbito de nuestro estudio. No por esto deja de ser uno de los puntos más importantes si se quisiera pasar a producción esta herramienta.

En cuanto a posteriores pasos, si a alguien le ha atraído el estudio de los procesadores de dinámica y la aplicación de filtros sobre la voz con vistas a mejorar la inteligibilidad al mismo nivel que al autor de este estudio, propondría aumentar el número de métricas a estudiar, y quizás se podría pensar, si se tuvieran medios para hacerlo, en el entrenamiento de una red neuronal, que en función del análisis del habla del interlocutor configurase las constantes de tiempo.

Capítulo 6 - Conclusions and future work

We have thus managed to develop the Improving Streaming tool, using the compressor of (Rudrich, 2019) and the high-pass filter of (Alermi, 2018). Two experiments have been carried out recording the broadcasts through Twitch, trying to expand the conclusions of the first by executing the second, in the way described in the previous Chapter. In both experiments we have found a direct relationship between the growth of the STOI and that of the compression on the filtered signal by the predictors. In fact, compressors are complex dynamics processors, because they have to deal with complex signals, and their configuration has to be adapted to the type of signal that passes through them, being very difficult to get closer to the optimum point by means of a preset in the case of the spoken signal, before the variety of its typology. We believe that this plugin can contribute and help, at the very least, to improve reception at a uniform volume, for announcers and professionals who work with software broadcasting programs that are compatible with VST technology. Although we have not obtained conclusive results for the above reasons, the Improving Streaming tool works very well offering what we were looking for: a dynamics processor and semi-automatic signal filter that anyone, without much computer knowledge, could configure and use. .

As a conclusion we could add that we have contributed something to the GNU world, as it was discussed since the beginning of the project, since it is available under this license and from now on in the following github repository: <https://github.com/ealcober/Improving-Streaming.git>. We have generated a self-configurable software, which allows a treatment adapted to each type of voice (except for the Threshold and the output level, whose configuration has not been automated and has been left in the hands of the user), allowing to apply different degrees of compression without degrading the original signal, unless the high pass filter is activated in the event that you want to transmit spoken voice with less “noise” in low frequencies, for which we consider the initial objectives of the project to be fulfilled. If the user does not know or does not want to adjust the threshold, the option to automate the instant configuration of this parameter with a simple tick is also included.

We would also have liked to invest the time necessary to optimize the performance and the order of complexity of the real-time processing function, but for the intelligibility study we focused on the fulfillment of the objective of the work, leaving this outside the scope of our study. This does not mean that it ceases to be one of the most important points if this tool were to be put into production.

As for further steps, if someone has been attracted to the study of dynamics processors and the application of filters on speech with a view to improving intelligibility to the same level as the author of this study, I would propose increasing the number of metrics to study, and perhaps one could think, if there were means to do so, in the training of a neural network, which, based on the analysis of the interlocutor's speech, would configure the time constants.

BIBLIOGRAFÍA

- Adecco, I. (2019-2020). *Informe Infoempleo Adecco*. Pozuelo de Alarcón, Madrid: THE ADECCO GROUP.
- Austerberry, D. (2005). *The Technology of Video and Audio Streaming*. Focal Press.
- CFI. (s.f.). *Cursos de Formación en Informática*. Recuperado el 01 de 06 de 2019, de <http://cursosinformatica.ucm.es>
- Crees H. Taal, R. C. (2010). *A Short Time Objective Intelligibility Measure for Time-Frequency Weighted noisy speech*. Delft, The Netherlands: Delft University of Technology, Signal information & Processing Lab.
- es.wikipedia.org. (01 de Marzo de 2021). Obtenido de Wikipedia: <https://es.wikipedia.org>
- Jianfen Ma, Y. H. (2009). *Objective measures for predicting speech intelligibility in noisy conditions based on new band importance functions*. Shanxi, 030024, China.: College of Computer Engineering and Software, Taiyuan University of Technology,.
- Rudrich, D. (2019). *SimpleCompressor*. Graz, Austria: GitHub, <https://github.com/DanielRudrich/SimpleCompressor>.

ÍNDICE DE FIGURAS

Ilustración 1, Índice de crecimiento de las descargas de aplicaciones de videoconferencia durante la semana del 15 al 21 de Marzo del 2020	1
Ilustración 2, Resultado informe Streamlabs (forbes) 2020 Q2, Q3 (https://streamlabs.com/content-hub/post/streamlabs-and-stream-hatchet-q3-2020-live-streaming-industry-report)	5
Ilustración 3, Canal SSL 500	6
Ilustración 4, señal mal preparada para ser comprimida	7
Ilustración 5, Eliminando las componentes carentes de información (zona marcada con una X) mediante el filtro paso altos, o high pass filter, podemos comprimir las fundamentales.....	8
Ilustración 6, Ficha temática dedicada a la compresión (McCormick, 1996)	10
Ilustración 7, Compresor comercial WAVES software, imitación de una reconocida unidad Analógica de alta gama, el API-2500 de SSL.....	11
Ilustración 8, Función de Transferencia de un compresor	12
Ilustración 9, Cómo afecta el ataque de un compresor.....	13
Ilustración 10, Cómo afecta el release de un compresor	13
Ilustración 11, Aportación del parámetro Knee a la configuración del compresor	14
Ilustración 12, Voz sin comprimir vs voz comprimida con un umbral de -15dB: http://alternativesilence.blogspot.com/2011/09/un-tutorial-mas-de-compresioncompressio.html	14
Ilustración 13, Análisis en tiempo real de voz masculina con headset Plantronics mediante plug-in Steinberg Waves F6-RTA Stereo.....	16
Ilustración 14, Señal de un instrumento desnaturalizada por el reductor de ruido automático de Microsoft Teams	17
Ilustración 15, Interfaz del compresor incorporado en OBS studio, LTS.....	18

Ilustración 16, Interfaz del compresor incorporado en OBS studio, LTS.....	20
Ilustración 17, El complicado abanico de plugins que ofrece Reaper.	21
Ilustración 18, ReaEQ, de Reaper	22
Ilustración 19, ReaCOMP, de Reaper	22
Ilustración 20, Ventana de "nuevo proyecto" en Juce al iniciar el programa.	23
Ilustración 21, Ventana de proyecto recién creado listo para abrirse en Visual Studio 2019. A la izquierda están los archivos generados por Juce al elegir la creación de un audio plug-in. En la parte superior está el selector de IDE que usaremos para editarlos y compilar el proyecto.....	24
Ilustración 22, Constructor y destructor del Procesador de Audio.....	25
Ilustración 23, Liberación de los punteros utilizados por el Filtro paso Altos y reseteo del Compresor para liberar los recursos asociados a ambas estructuras.	26
Ilustración 24, Función de proceso en tiempo real del buffer de Audio.	27
Ilustración 25, Función prepareToPlay, que reserva los recursos utilizados en processBlock.	27
Ilustración 26, Aspecto final del plugin Improving Streaming desarrollado durante este TFG.....	28
Ilustración 27, Compresor original de Daniel Rudrich, 2019, GNU, Git-Hub available.	29
Ilustración 28, función original processBlock de Daniel Rudrich.....	30

ÍNDICE DE TABLAS

Tabla 1, mejoras de audio disponibles en plataformas de comunicaciones & teletrabajo	4
Tabla 2, Componentes frecuenciales de la Voz	15
Tabla 3, Ejemplo de resultado con el comando dwdiff	36
Tabla 4, Presets del procesador en el primer experimento	38
Tabla 5, Resultados primer experimento	39
Tabla 6, Parámetros del segundo experimento.....	43
Tabla 7, resultados promedio de los 4 interlocutores en el segundo experimento.	45

DEDICATORIA

A todos los que han descubierto que la búsqueda de la Virtud trae en sí misma su propia recompensa.

AGRADECIMIENTOS

Quiero agradecer a mis tutores, Don Miguel y Don Jaime, la confianza depositada y el tiempo y recursos invertidos durante la tutorización del TFG. A mi familia por ayudarme e insistirme en que terminase la Universidad, y apoyarme de todas las formas posibles y, a mis hermanos y hermanas, por creer en mí cuando nadie más lo hacía.

